

# Représentations parcimonieuses des signaux

---

Cours ST

Stéphane Rossignol – stephane.rossignol@centralesupelec.fr

3 février 2025

Les slides sont disponibles ici

Teams – > ST7-SEP\_RepParci\_2024-2025 – > Fichiers – >  
Supports de cours – >

Attention : version qui ne va sans doute pas évoluer

# Plan

- ▶ **Partie 1 : Introduction (qu'est-ce que la parcimonie ?)**
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ Partie 7 : Conclusion

## Mes recherches sont liées à ce cours

- ▶ Études à l'Université : DUT EEEII, licence et maîtrise d'électronique, DEA signal télécom image radar (option radar), DEA instrumentation en sciences de l'univers
- ▶ Service militaire en tant que scientifique du contingent – détection de masses ferromagnétiques de 300 tonnes se déplaçant sous l'eau
- ▶ Thèse finie en juillet 2000 ; thèse sur la segmentation et l'indexation des signaux sonores musicaux, faite à l'IRCAM et ici, en partenariat avec France Télécom-Orange
- ▶ Modification de la prosodie d'un acteur (Depardieu parlant anglais dans Vatel de Roland Joffé)
- ▶ Postdoc (2000-2001) : modélisation du vibrato dans une université néerlandaise (⇒ aide à la composition, MPEG-7)
- ▶ Postdoc (2001-2005) : mise en place de systèmes de dialogue dans une université néerlandaise (projet européen COMIC)

## Mes recherches sont liées à ce cours

- ▶ Postdoc (2006) : mise en place d'un système de reconnaissance de la parole (pour France Télécom-Orange)
- ▶ Postdoc (2007) : caractérisation de défauts de haut-parleurs sur la ligne de production (équipementier automobile)
- ▶ Enseignant-chercheur ici depuis 2008
  - ▶ recherches propres : traitement de la musique (et de la parole)
  - ▶ mise en place de systèmes de dialogue (projets européens CLASSIC, ALLEGRO, HILAIRE)
  - ▶ holophonie
  - ▶ cours concernant principalement le traitement du signal (SIG1, SIG2, RASS, analyse spectrale, traitement de la parole)
- ▶ Équipe Multispeech du LORIA à Nancy

# Où suis-je (normalement) ?

- ▶ au troisième étage, dans la partie LORIA
- ▶ bureau : B306
- ▶ téléphone : 03 87 76 47 73

# Organisation du cours

- ▶ Cours
  - ▶ il y a un syllabus ; je ne sais pas si j'aurai le temps de tout faire ; mais ce qui est important, c'est que vous fassiez des choses par vous-mêmes
  - ▶ en 2021, j'ai enregistré les cours (dites si vous voulez les récupérer)
- ▶ TL/TD
  - ▶ je donne du code en octave/matlab, mais si vous voulez tout faire en python, pas de problème
  - ▶ binômez-vous : éventuellement, je ferai un canal Teams par binôme, et lors des séances de TL/TD, je passerai canal par canal (une dizaine de binômes, ça donne 8-9 minutes par canal) ⇒ ça a bien marché ces dernières années
- ▶ Examen
  - ▶ une présentation, binôme par binôme, d'environ une demi-heure, sur ce que vous allez faire en TL/TD

## Qu'est-ce que la parcimonie ?

- ▶ Un son sinusoïdal (d'un diapason), enregistré à 44100 Hz, avec 2 octets par échantillon, pendant 10 secondes, ça prend presque 1 Mo (ou 441 mille nombres à virgule)
  - ▶ 

```
fe=44100;xx=cos(2*pi*440*[0 :1/fe :10]);audio-write('diap.wav',xx,fe,'BitsPerSample',16);
```
- ▶ Alors qu'il n'y a besoin que de 3 nombres à virgule pour coder un sinus : une amplitude, une fréquence, une phase (et même 2 nombres, car la phase n'est pas perçue par l'oreille)
  - ▶ le taux de compression possible est alors de près de 150000
- ▶ On essaie de faire ça : représenter les signaux numériques de façon plus parcimonieuse
- ▶ Bien sûr, le cas est extrême : il faut coder le temps de début et le temps de fin de la sinusoïde, le fade-in et le fade-out, les harmoniques éventuelles, etc.

# Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ **Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)**
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ Partie 7 : Conclusion

# La transformée de Fourier

- ▶ Projection sur l'espace des exponentielles complexes
  - ▶ les  $\exp(j2\pi ft)$  forment une base orthonormée de l'espace vectoriel des fonctions

Décomposition

Transformation inverse

$$x(t) = \int_{-\infty}^{+\infty} X(f) \exp(+j2\pi ft) df$$

$$x[k] = \int_0^1 X(\nu) \exp(+j2\pi\nu k) d\nu$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \exp\left(+j2\pi \frac{nk}{N}\right)$$

Produit scalaire

Transformation

$$X(f) = \int_{-\infty}^{+\infty} x(t) \exp(-j2\pi ft) dt$$

$$X(\nu) = \sum_{k=-\infty}^{+\infty} x[k] \exp(-j2\pi\nu k)$$

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(-j2\pi \frac{nk}{N}\right)$$

## Transformée de Fourier (TF) des signaux discrets

Il s'agit de relier la représentation spectrale  $X(f)$  du signal analogique à la représentation spectrale  $\mathbf{X}(\nu)$  du signal discret  $\Rightarrow$  attention : c'est assez laborieux.

- ▶ définition de la TF analogique :

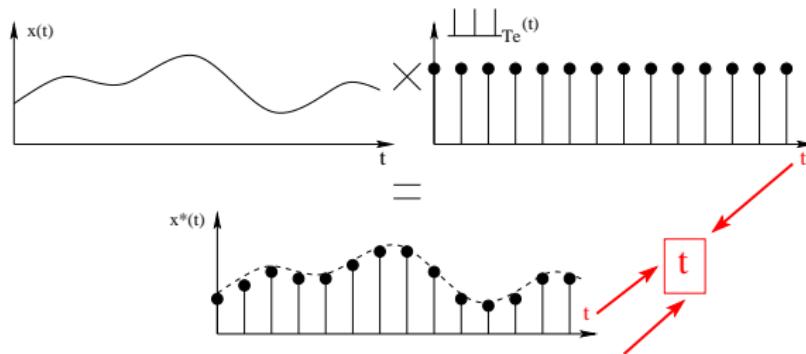
$$X(f) = \int_{-\infty}^{+\infty} x(t) \exp(-2\pi jft) dt$$

- ▶ définition de la TF discrète :

$$\mathbf{X}(\nu) = \sum_{k=-\infty}^{+\infty} x[k] \exp(-2\pi j\nu k)$$

## Signal échantillonné

- ▶ Signal échantillonné :  $x^*(t) = x(t) \times \prod_{T_e}$



- ▶ **Attention :** le signal échantillonné  $x^*(\textcolor{red}{t})$  n'est pas le signal discret  $x[k]$  !
  - ▶ le signal échantillonné est un signal **fictif** analogique ( $x^*(\textcolor{red}{t})$ )
  - ▶ qui prend des valeurs non nulles aux instants d'échantillonnage
  - ▶ et est nul partout ailleurs

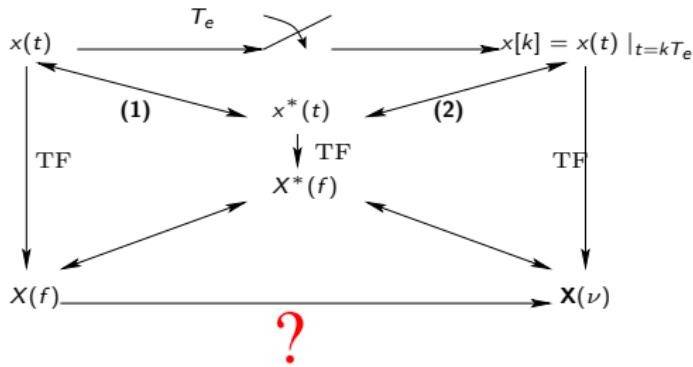
## Relation spectrale (1/2)

- ▶ Il faut mettre en relation  $X(f)$  et  $\mathbf{X}(\nu)$
- ▶ et, pour cela, le passage par le *signal échantillonné*  $x^*(t)$
- ▶ et par le petit schéma ci-dessous aide :

$$x^*(t) = \sum_{k=-\infty}^{+\infty} x(kT_e) \delta(t - kT_e)$$

$$x^*(t) = x(t) \sum_{k=-\infty}^{+\infty} \delta(t - kT_e) \quad (1)$$

$$x^*(t) = \sum_{k=-\infty}^{+\infty} x[k] \delta(t - kT_e) \quad (2)$$



## Relation spectrale (2/2)

$$x^*(t) = x(t) \sum_{k=-\infty}^{+\infty} \delta(t - kT_e) \quad (1)$$

$$x^*(t) = \sum_{k=-\infty}^{+\infty} x[k] \delta(t - kT_e) \quad (2)$$

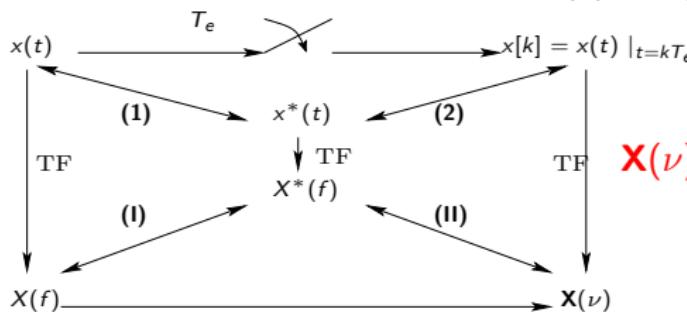
$$X^*(f) = X(f) * \frac{1}{T_e} \sum_{n=-\infty}^{+\infty} \delta\left(f - \frac{n}{T_e}\right)$$

$$X^*(f) = \int_{-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} x[k] \delta(t - kT_e) \exp(-2\pi jft) dt$$

$$X^*(f) = \frac{1}{T_e} \sum_{n=-\infty}^{+\infty} X\left(f - \frac{n}{T_e}\right) \quad (\text{I})$$

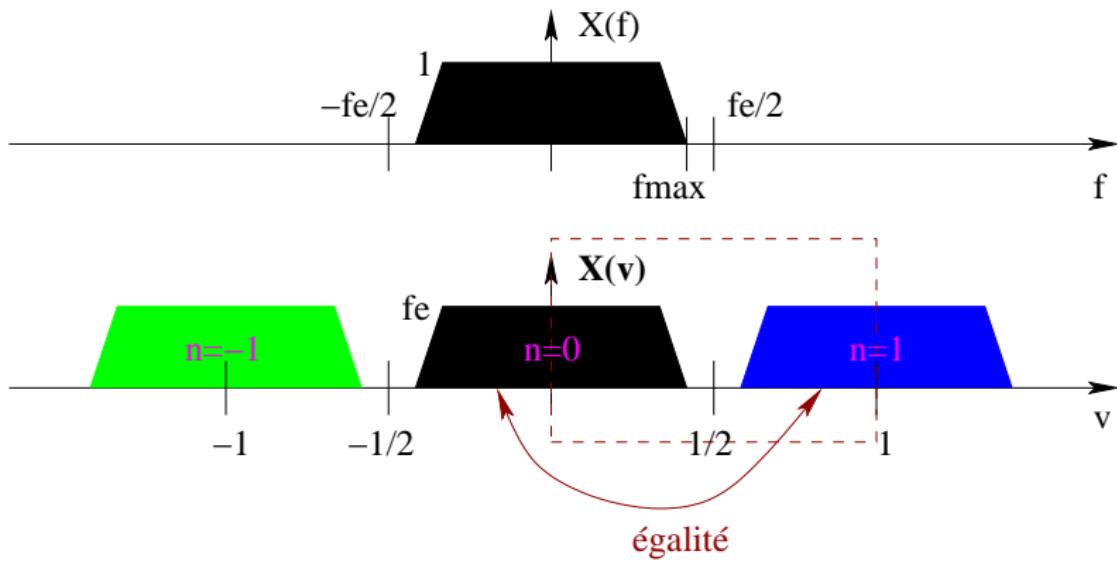
$$X^*(f) = \sum_{k=-\infty}^{+\infty} x[k] \exp(-2\pi jk f T_e) \quad \nu = f T_e$$

$$X^*(f) = \mathbf{X}(\nu) \Big|_{\nu=fT_e} \quad (\text{II})$$

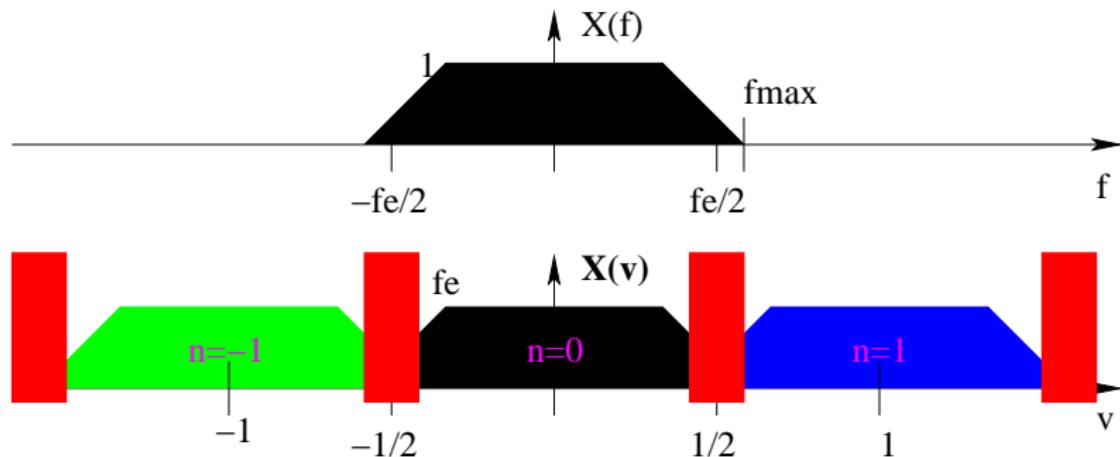


$$\mathbf{X}(\nu) = f_e \sum_{n=-\infty}^{+\infty} X\left(f - \frac{n}{T_e}\right) \Big|_{f=\nu f_e}$$

## Exemple 1



## Exemple 2



⇒ les zones en rouge indiquent que pour ces fréquences il y a du **recouvrement spectral** : pour ces fréquences de l'information est **détruite, définitivement perdue**

## Théorème de Shannon

- ▶ Pour que l'échantillonnage conserve l'information spectrale sans la déformer,
- ▶ si le spectre du signal est borné par  $f_{\max}$ ,
- ▶ la fréquence d'échantillonnage  $f_e$  doit vérifier :

$$f_e > 2f_{\max}$$

## Horizon fini : $N$ échantillons/double échantillonnage

- ▶ Le signal **temporel** est composé de  $N$  échantillons
- ▶ Les TF considérées jusqu'à présent sont **continues** ; cependant, on ne peut pas physiquement calculer la TF pour une infinité de  $\nu$  : il faut donc choisir judicieusement un nombre fini de  $\nu_i$ . Au sujet de ces  $\nu_i$  :
  - ▶ comment sont-elles disposées sur l'axe des fréquences ?  
⇒ a priori, régulièrement
  - ▶ et, dès lors, pour ne pas perdre d'information, quel écart maximum choisir entre deux  $\nu_i$  successifs, comment échantillonner l'axe des fréquences ?  
⇒ on peut montrer que  $N$  fréquences réduites  $\nu_i$  suffisent

## Horizon fini – échantillonnage des fréquences

- ▶ Discrétisation des fréquences (pour obtenir la TFD) :

$$\nu = \left[ 0 \quad \frac{1}{N} \quad \frac{2}{N} \quad \dots \quad \frac{N-1}{N} \right]$$

- ▶ C'est intuitif : si  $N$  échantillons suffisent à décrire complètement le signal dans le domaine temporel, il n'y a pas de raison particulière pour qu'il en faille plus que  $N$  dans le domaine fréquentiel pour faire la même chose, ou qu'avec moins que  $N$  échantillons ça marche
- ▶ Ça se montre si on considère que la TF de la TF redonne les échantillons du signal d'origine (à une inversion de l'axe des temps près) ; donc, on part de  $N$  points, on tombe sur  $N$  points, et on retombe sur les  $N$  points d'origine
- ▶ Note : les  $X_N[k]$  sont les poids de composantes harmoniques tronquées intervenant dans le signal

## Horizon fini – échantillonnage des fréquences

**TFD directe :**

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] \exp\left(-2\pi j \frac{nk}{N}\right)$$

pour  $k \in [0, N - 1]$ ,  $\nu \in \left[0, \frac{N - 1}{N}\right]$

$$X_N[k] = X_N(\nu) \Big|_{\nu=\frac{k}{N}}$$

**TFD inverse :**

$$x_N[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_N[k] \exp\left(2\pi j \frac{nk}{N}\right)$$

pour  $n \in [0, N - 1]$

$$x[n] = \frac{1}{N} \overline{\text{TFD} [\overline{X[k]}]}$$

## Algorithmes de TFD – FFT

- ▶ la TFD possède des algorithmes **rapides** (accélérés par rapport à la formule donnée dans le transparent précédent)
  - ▶ FFT (**Fast** Fourier Transform) ⇒ voir matlab
  - ▶ TFR (Transformée de Fourier **Rapide**) [terme francophone pour la FFT]
- ▶ si  $N = b^r$  (où  $N$ ,  $b$  et  $r$  sont des entiers)
  - ▶ il existe des algorithmes (itératifs) spécifiques pour chaque base  $b$
  - ▶ le plus connu et le plus utilisé étant celui pour  $b = 2$  : Radix-2 (de Cooley-Tuckey, 1965) ⇒ on passe de  $n_1 = N^2$  opérations (+,  $\times$ ) à  $n_2 = N \log_2 N$  (exemple :  $N = 1024 \Rightarrow n_1/n_2 \simeq 100$  ;  $N = 8192 \Rightarrow n_1/n_2 \simeq 630$ )
  - ▶ de plus, parallélisation possible du calcul
- ▶ attention : si  $N$  est un nombre premier (3, 5, 7, 11, 13...), il n'y a pas moyen d'accélérer les calculs

## Limites de l'analyse spectrale formalisées par Gabor

- ▶ Principe d'incertitude de Gabor :

$$\overline{\Delta t}^2 \overline{\Delta f}^2 \geq \frac{1}{16\pi^2} \text{ (inégalité de Gabor)}$$

$$\overline{\Delta f}^2 = \frac{\int_{-f_{max}}^{f_{max}} f^2 |X(f)|^2 df}{\int_{-f_{max}}^{f_{max}} |X(f)|^2 df} = \frac{A}{B}$$

⇒ Support spectral moyen d'un signal autour de  $f = 0$  (0 est nécessairement la fréquence moyenne si le signal est réel)

$$\overline{\Delta t}^2 = \frac{\int_{-\infty}^{+\infty} (t - t_0)^2 x^2(t) dt}{\int_{-\infty}^{+\infty} x^2(t) dt} = \frac{C}{D} \text{ avec } t_0 = \frac{\int_{-\infty}^{+\infty} tx^2(t) dt}{\int_{-\infty}^{+\infty} x^2(t) dt}$$

⇒ Durée moyenne d'un signal autour de  $t_0$ .

## Fenêtrage – Résolution spectrale et troncature (1/9)

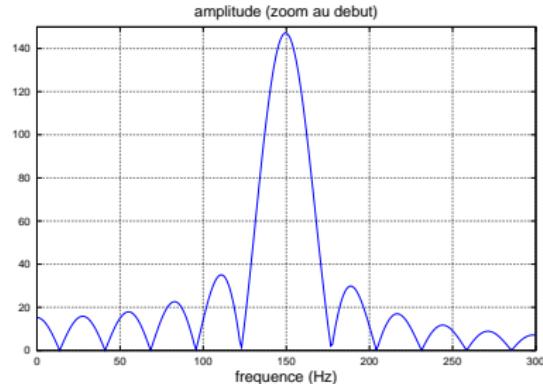
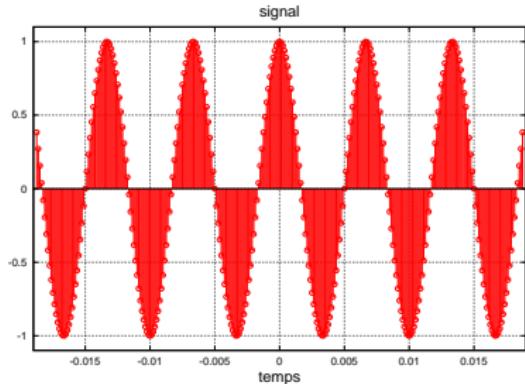
- ▶ On n'observe le signal que sur un temps fini :  $N$  points
- ▶ On a vu que ça équivalait à multiplier le signal par une fenêtre rectangulaire (porte)
- ▶ Multiplier par la porte de largeur  $N$  (ou  $T$ ) dans le domaine temporel revient à convoluer par un sinus cardinal dans le domaine fréquentiel
- ▶ Le lobe principal du sinus cardinal est de largeur  $2/N$  (ou  $2/T$ )
- ▶ La résolution, c'est la distance en fréquence (en Hz) minimale entre deux sinusoïdes de telle façon qu'elles soient séparables par Analyse Spectrale (séparables, ici, veut dire que leurs lobes principaux ne se recouvrent pas, qu'il y a un creux notable entre les deux lobes principaux)

## Résolution spectrale et troncature (2/9)

- ▶ Résolutions spectrales comparées :
  - ▶ si la fenêtre rectangulaire est utilisée, on a  $\frac{1}{T}$ ,  $T$  étant l'horizon d'observation
  - ▶ si la fenêtre de Hanning est utilisée, on a  $\frac{2}{T} \Rightarrow$  c'est moins bon, mais **ce défaut est nettement compensé par les qualités du fenêtrage de Hanning démontrées par les exemples de ce cours !**  
⇒ il faut trouver un compromis, comme d'habitude

## Résolution spectrale et troncature (3/9) – Exemples

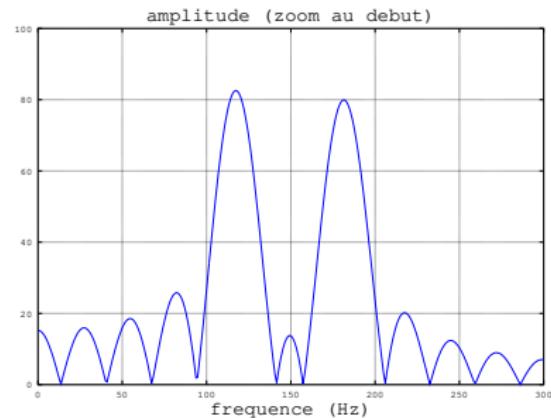
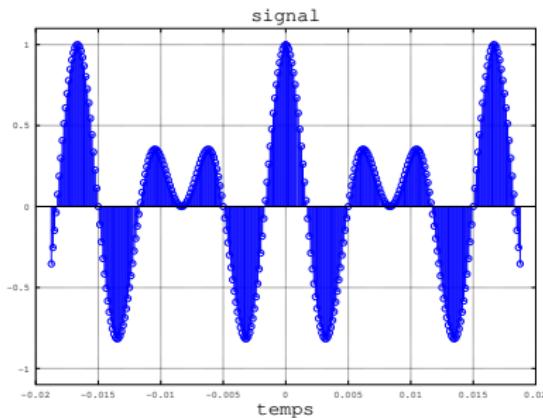
Signal : un cosinus tronqué, de fréquence 150 Hz (voir resolu1.m)



Si on observe le lobe central (ou principal), on constate que la TF est rendue floue par la troncature (effets de bords) ; et les lobes secondaires sont d'amplitude importante

## Résolution spectrale et troncature (4/9) – Exemples

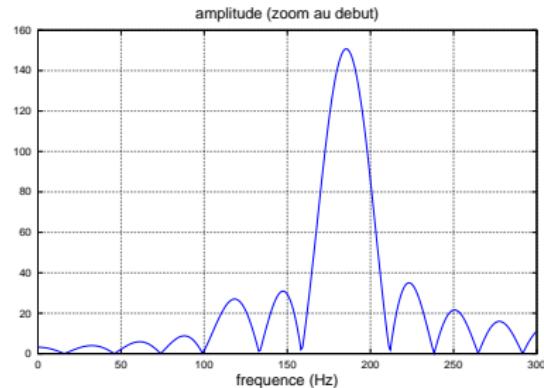
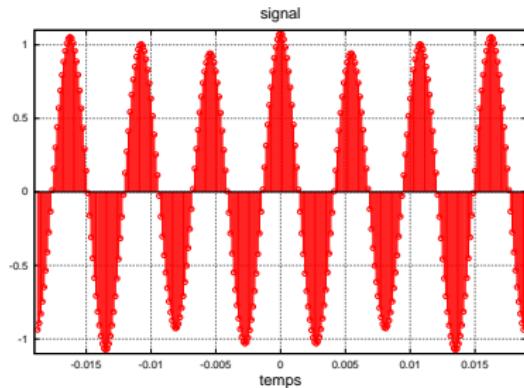
Signal : 2 cosinus tronqués, de fréquence 115 Hz et 185 Hz, de même amplitude



Si les fréquences sont trop proches, les lobes centraux se recouvrent ; une meilleure résolution est obtenue si le lobe principal est plus étroit, mais alors les problèmes dus aux lobes secondaires deviennent trop importants ⇒ voir resolu2.m

## Résolution spectrale et troncature (5/9) – Exemples

Signal : 2 cosinus tronqués, de fréquence 115 Hz et 185 Hz, d'amplitude 0.08 et 1 (voir resolu3.m)



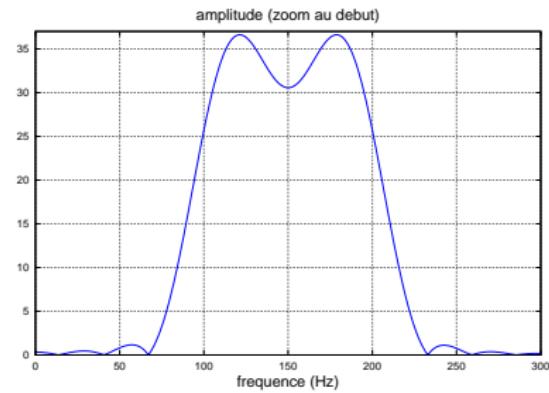
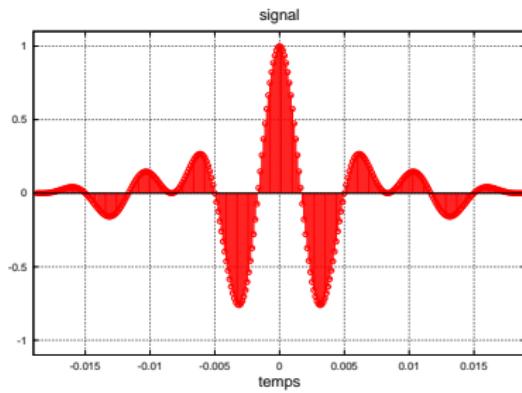
Si les amplitudes sont trop différentes, le lobe central de la plus petite est caché dans les lobes secondaires de l'autre

## Résolution spectrale et troncature (6/9) – Exemples

- ▶ Idéalement, pour améliorer la résolution spectrale (en amplitude et en fréquence), il faudrait élargir l'horizon d'observation, ce qui n'est pas possible pour des raisons de stationnarité
- ▶ Ou alors il faut pondérer par une autre fenêtre que la fenêtre rectangulaire
  - ▶ pour minimiser les lobes secondaires, il faut adoucir les bords, mais alors la largeur du lobe principal augmente
  - ▶ pour amincir le lobe principal, il faudrait rendre les bords encore plus abrupts par rapport au centre de la fenêtre que ce qu'on a pour la fenêtre rectangulaire, mais alors l'amplitude des lobes secondaires augmente
  - ▶ ⇒ en fait, il n'y a pas de solution idéale : il faut faire un compromis

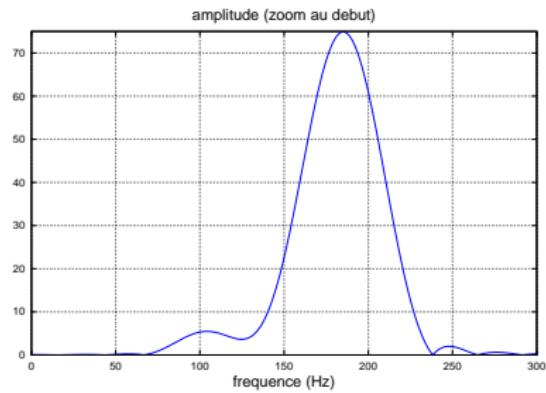
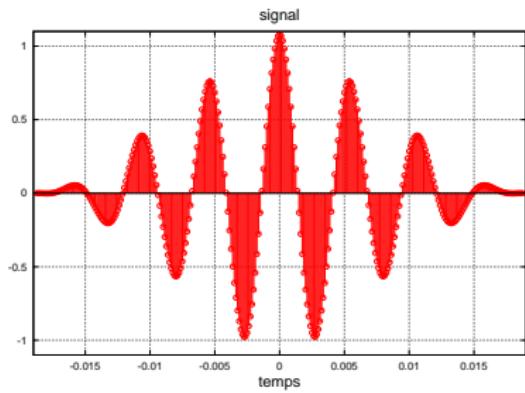
## Résolution spectrale et troncature (7/9) – Exemples

Signal : 2 cosinus tronqués, de fréquence 120 Hz et 180 Hz, de même amplitude ; pondération par Hanning (voir resolu4.m)



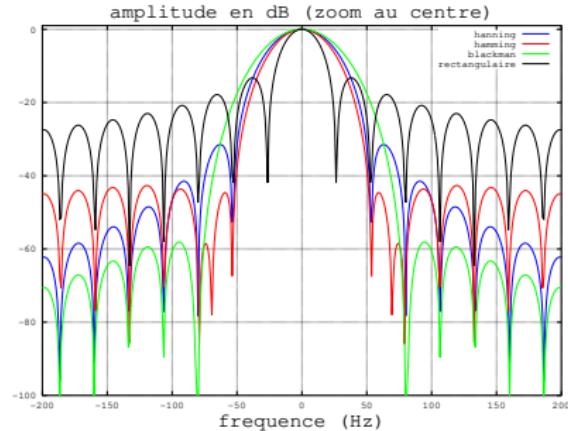
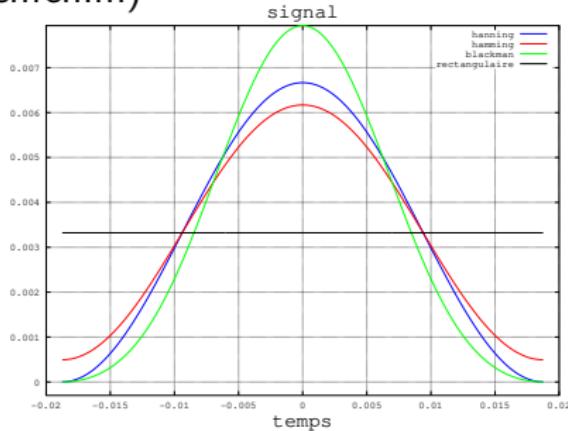
## Résolution spectrale et troncature (8/9) – Exemples

Signal : 2 cosinus tronqués, de fréquence 115 Hz et 185 Hz,  
d'amplitude 0.08 et 1; pondération par Hanning (voir resolu5.m)



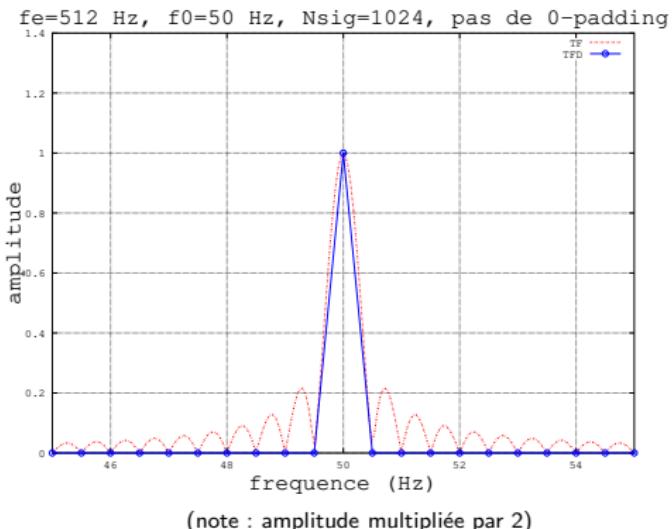
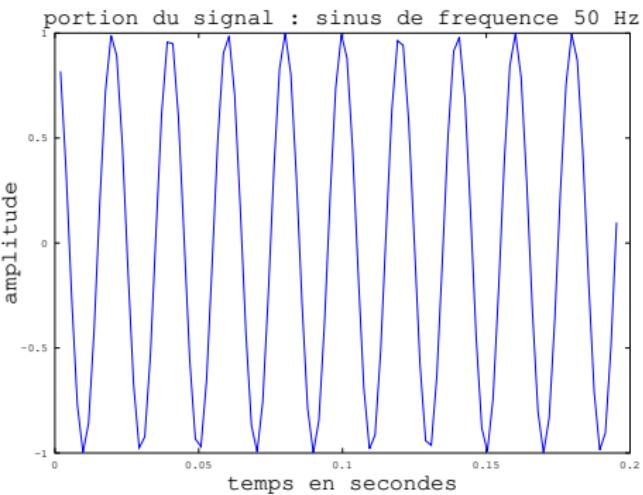
## Résolution spectrale et troncature (9/9) – Exemples

Fenêtres les plus courantes : Hanning, Hamming, Blackman (voir fenfen.m)



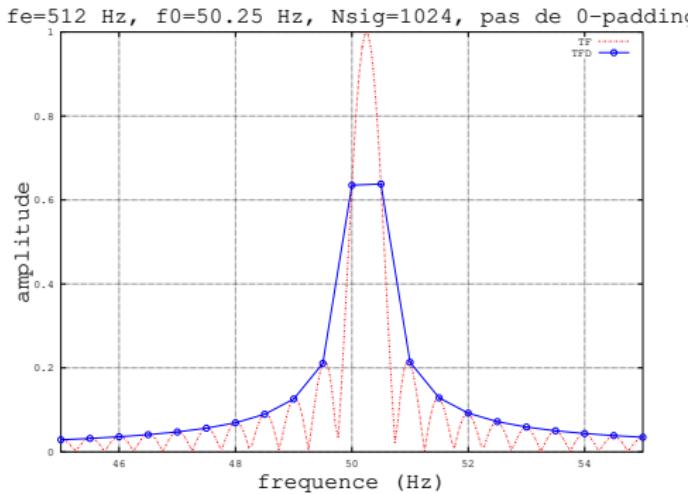
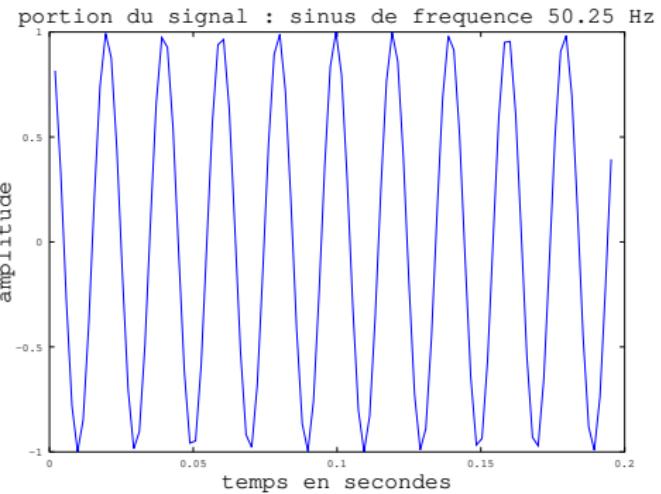
Point commun : meilleure résolution en amplitude/moins bonne résolution spectrale

## Effet de l'échantillonnage fréquentiel (1/3)



Si  $\nu_0 = k/N$ ,  $\nu_0$  est un point de calcul de la TFD

## Effet de l'échantillonnage fréquentiel (2/3)

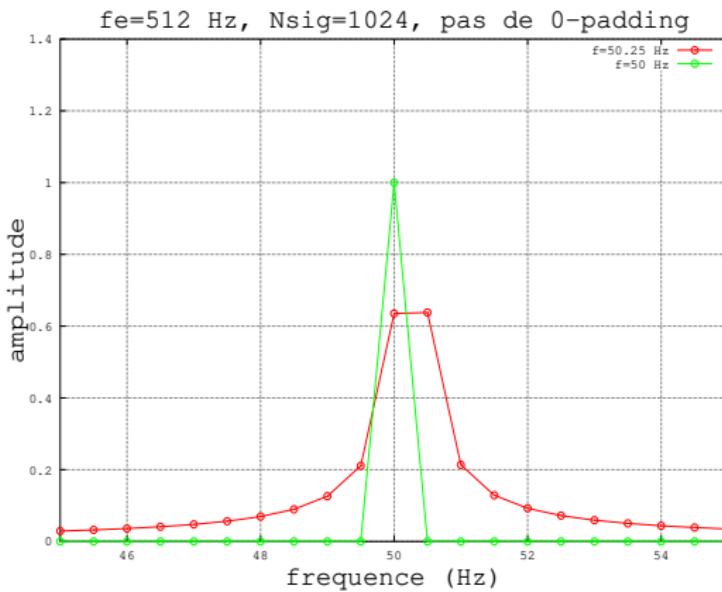


Si  $\nu_0 \neq k/N$ , incertitude proportionnelle à  $1/N$  sur la fréquence, et incertitude sur l'amplitude

Note : le 0-padding et le fenêtrage aident à résoudre ça

## Effet de l'échantillonnage fréquentiel (3/3)

Comparaison entre les 2 spectres précédents :

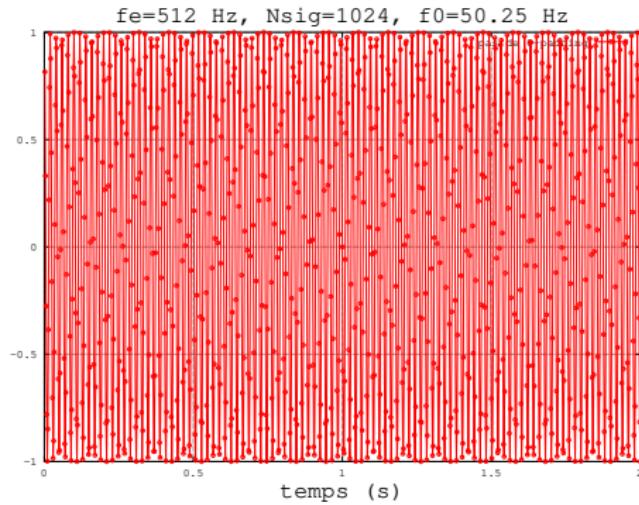


## Effet du zéro-padding (1/4)

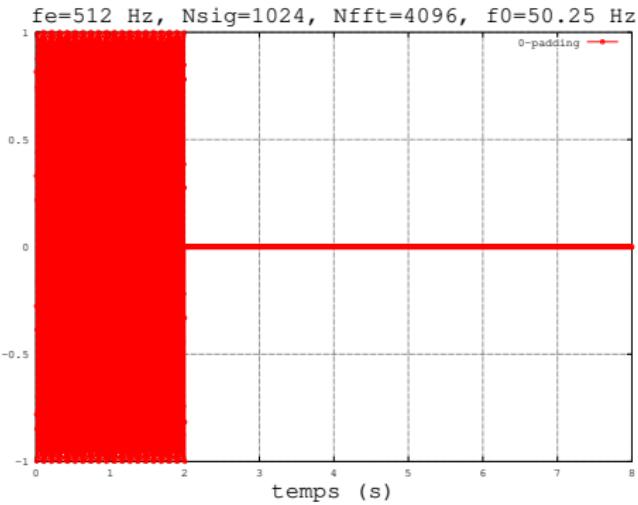
- ▶ Définition : il s'agit d'allonger un signal de longueur  $P$  échantillons utiles en ajoutant  $M$  zéros à la fin, pour obtenir un signal de taille  $N = P + M$
- ▶ Amélioration de la **visualisation/représentation** des résultats
  - ▶ ⇒ **attention** : on ne prend pas plus d'échantillons du signal utiles, donc le zéro-padding (bourrage de zéros, en français) n'apporte aucune information supplémentaire (simple interpolation des points de la FFT de longueur  $P$ )
  - ▶ en fait, on échantillonne la TF avec un pas de  $1/N$ , et plus avec un pas de  $1/P$
- ▶ Sert aussi (et c'est important !) à se mettre dans la configuration  $N = 2^r$ , avec  $r$  entier

## Effet du zéro-padding (2/4) (voir exemples\_nplusi.m)

Signal sans 0-padding :

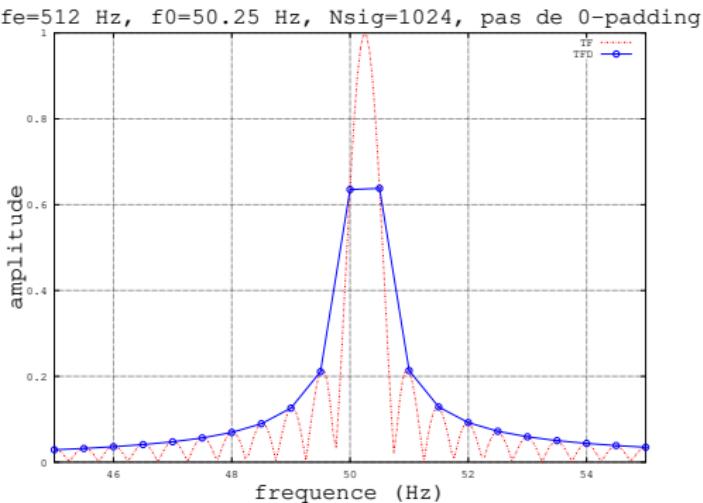


Signal avec 0-padding :

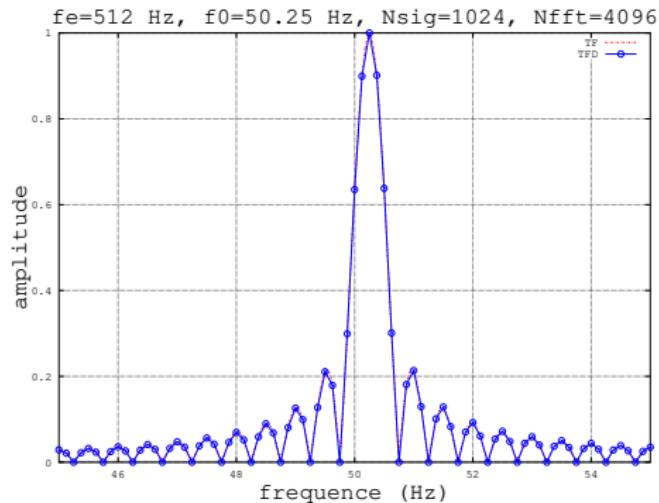


## Effet du zéro-padding (3/4) (voir exemples\_nplusi.m)

FFT du signal sans 0-padding

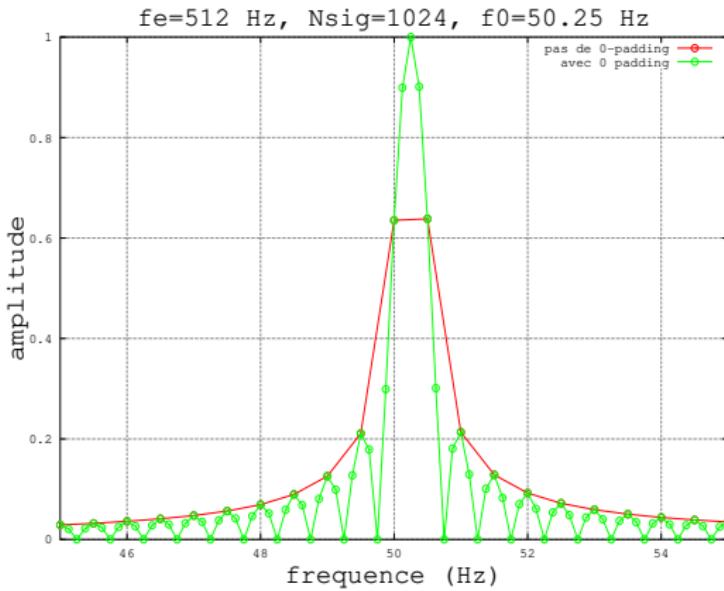


FFT du signal avec 0-padding



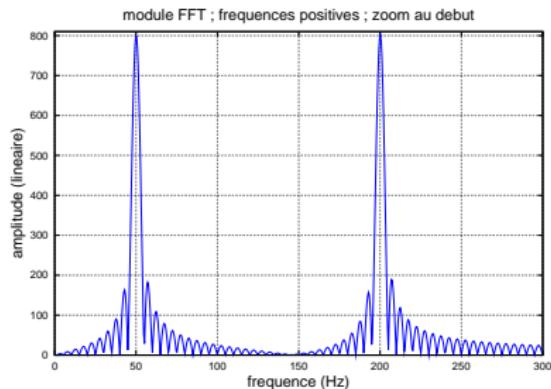
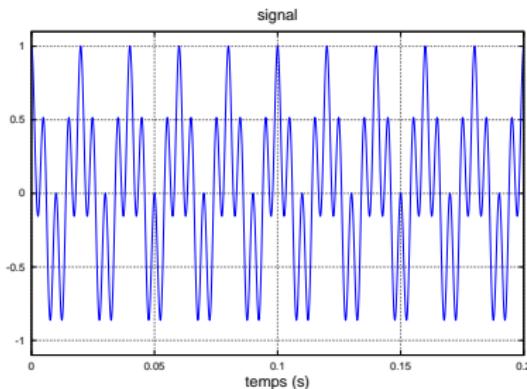
## Effet du zéro-padding (4/4) (voir exemples\_nplusi.m)

Comparaison des 2 FFT précédentes



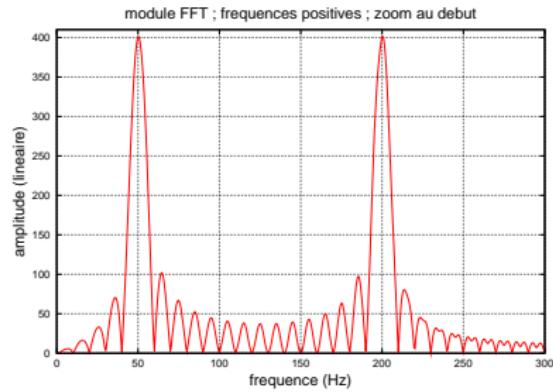
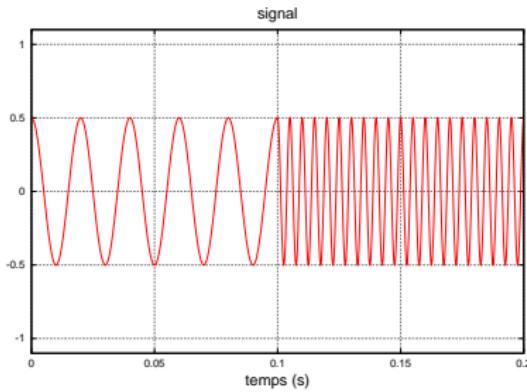
## Signaux non-stationnaires – Exemple 1

- ▶ 2 cosinus présents tout le temps ensemble
- ▶  $x(t) = 0.5 \cos(2\pi 50t) + 0.5 \cos(2\pi 200t)$
- ▶ le signal n'est pas stationnaire car il est tronqué ( $t \in [0, 0.2]$ )



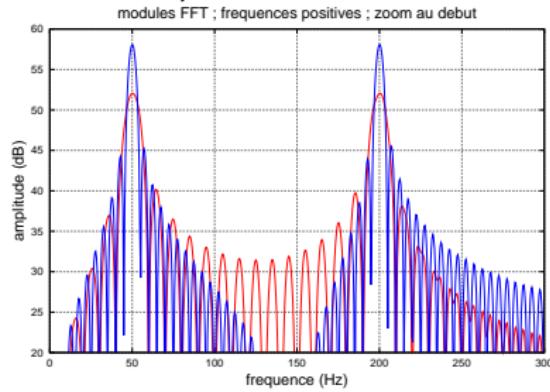
## Signaux non-stationnaires – Exemple 2

- ▶ 2 cosinus présents successivement
- ▶  $x(t) = 0.5 \cos(2\pi 50t)$  pour  $t \in [0, 0.1]$  ;  
 $x(t) = 0.5 \cos(2\pi 200t)$  pour  $t \in [0.1, 0.2]$
- ▶ le signal n'est pas stationnaire : il est tronqué, et surtout il y a une transition



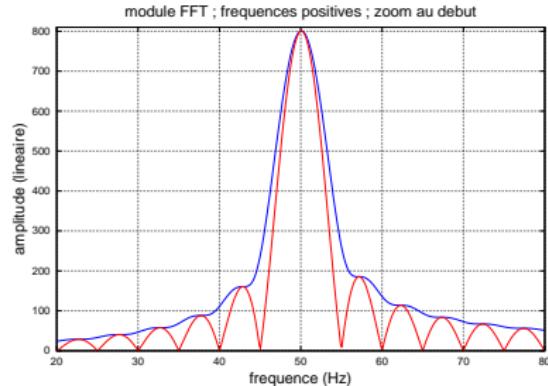
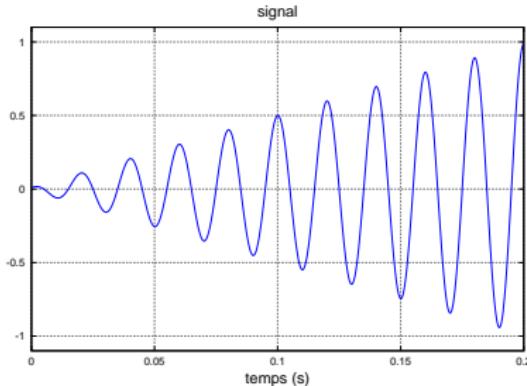
## Comparaison exemples 1 et 2

- ▶ on a les 2 raies aux mêmes endroits (50 Hz et 200 Hz)
- ▶ les lobes secondaires de l'exemple 2 sont plus grands, à cause du transitoire (il faut plus d'exponentielles complexes pour le modéliser)
- ▶ on sent bien que l'analyse de l'exemple 2 n'est pas absolument satisfaisante, de toute façon



## Signaux non-stationnaires – Exemple 3

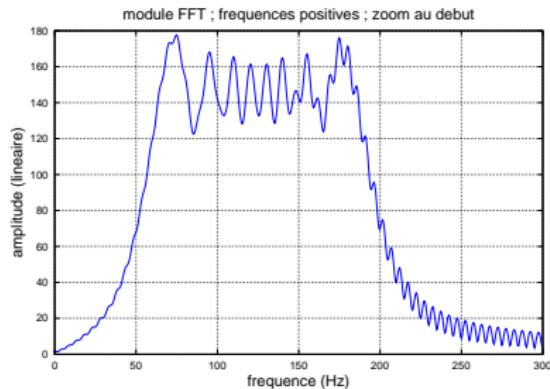
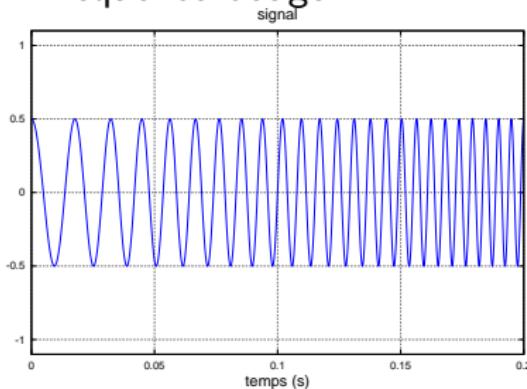
- ▶ 1 cosinus dont l'amplitude varie linéairement de 0.01 à 0.99
- ▶  $x(t) = A(t) \cos(2\pi 50t)$  pour  $t \in [0, 0.2]$
- ▶ signal pas stationnaire : il est tronqué, et surtout l'amplitude bouge



- ▶ on compare avec le spectre pour un cosinus d'amplitude fixe (rougeclair)
- ▶ on note l'élargissement du lobe principal

## Signaux non-stationnaires – Exemple 4

- ▶ 1 cosinus de fréquence variant linéairement de 50 à 200 Hz
- ▶  $x(t) = 0.5 \cos(\phi(t))$  pour  $t \in [0, 0.2]$
- ▶ le signal n'est pas stationnaire : il est tronqué, et surtout la fréquence bouge



- ▶ il est évident que l'analyse n'est pas très satisfaisante

## Transformée de Fourier à Court Terme (TFCT)

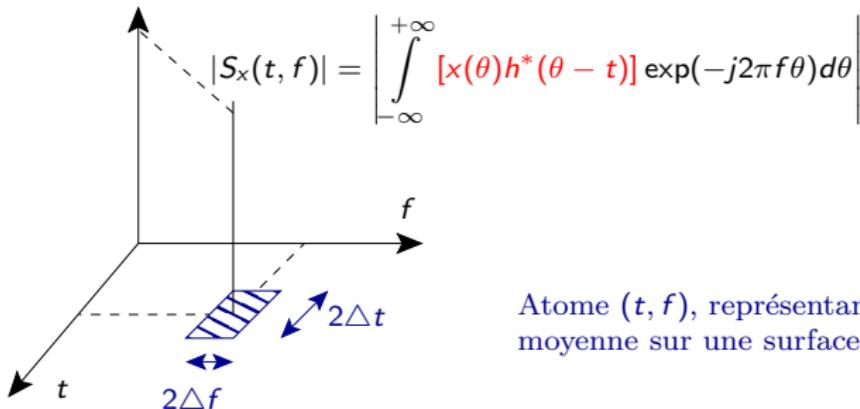
- ▶ Pour résoudre ce types de problèmes, Gabor proposa la transformation temps-fréquence suivante :

$$\text{TFCT} : S_x(t, f) = \int_{-\infty}^{+\infty} x(\theta) h^*(\theta - t) \exp(-j2\pi f\theta) d\theta$$

- ▶ qui récupère certaines propriétés de l'inégalité de Gabor
  - ▶ résolution espérée :  $\overline{\Delta t}^2 \overline{\Delta f}^2 \geq \frac{1}{16\pi^2}$
  - ▶ l'égalité étant atteinte pour la fenêtre gaussienne :  
$$h(\theta) = \exp(-\lambda\theta^2)$$
- ▶ En général :  $h(\theta)$  est centrée en 0 et à support temporel borné
- ▶ ⇒ On applique au signal, donc, en fait, une fenêtre temporelle glissante (du type Hanning, Hamming, Blackman...)

## TFCT – Interprétation 1

- ▶ Fenêtre glissante appliquée sur le signal, puis TF
  - ▶ on applique une fenêtre centrée en  $t$  et de durée moyenne finie  $\pm \Delta t$
  - ▶ la TF de ceci donne la convolution de  $X(f)$  et de  $H(f)$ , c'est-à-dire  $X(f)$  lissée en moyenne sur un intervalle  $\pm \Delta f$

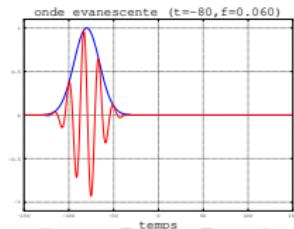
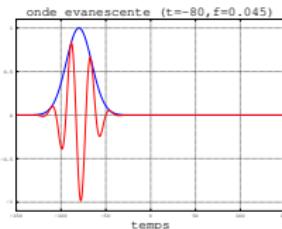
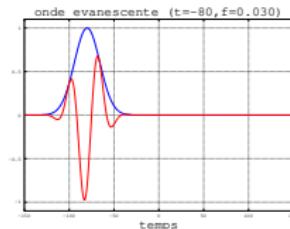
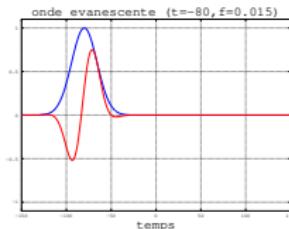
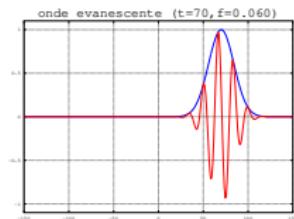
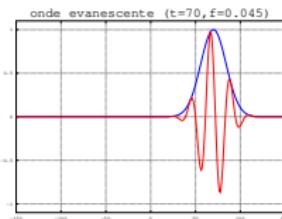
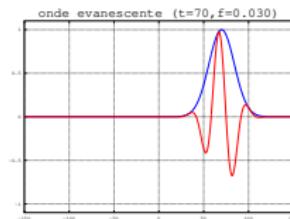
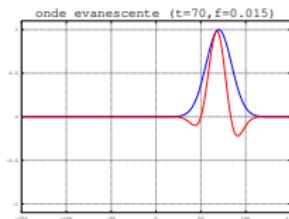


Atome  $(t, f)$ , représentant une moyenne sur une surface constante

## TFCT – Interprétation 2 – voir evanescentes.m

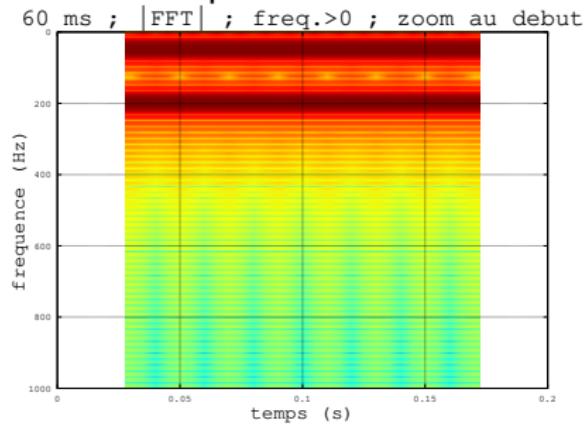
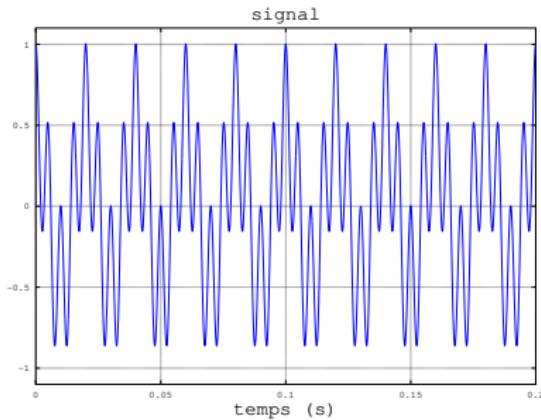
- ▶ Projection du signal sur des ondes évanescantes

$$S_x(t, f) = \int_{-\infty}^{+\infty} x(\theta) h^*(\theta - t) \exp(-j2\pi f\theta) d\theta$$

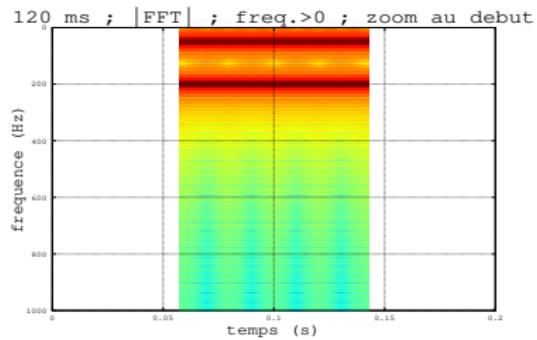
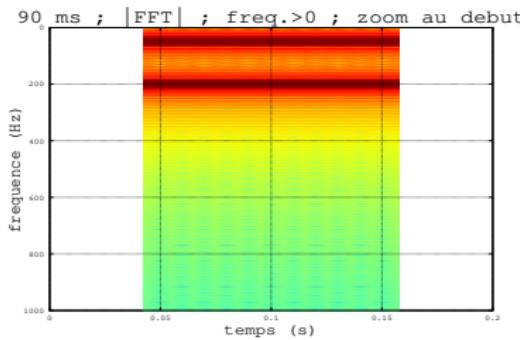
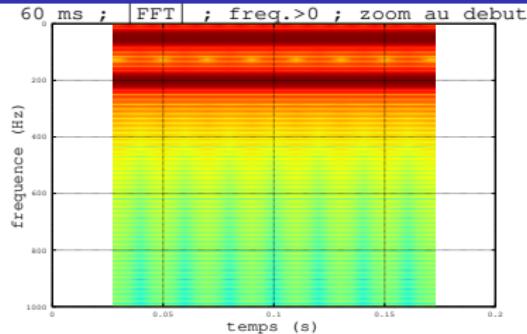
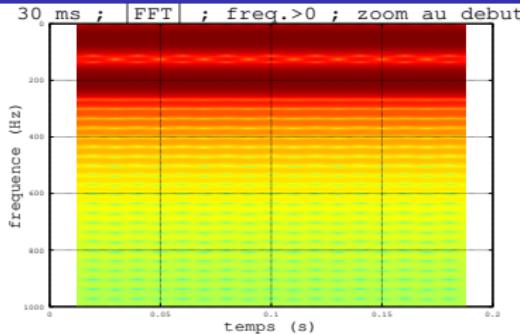


## Retour à l'exemple 1

- ▶ 2 cosinus présents tout le temps ensemble :  
 $x(t) = 0.5 \cos(2\pi 50t) + 0.5 \cos(2\pi 200t)$
- ▶  $h(\theta)$  est une fenêtre de Hanning large de 60 ms
- ▶ lecture de la TFCT : la couleur donne l'amplitude de la TFCT

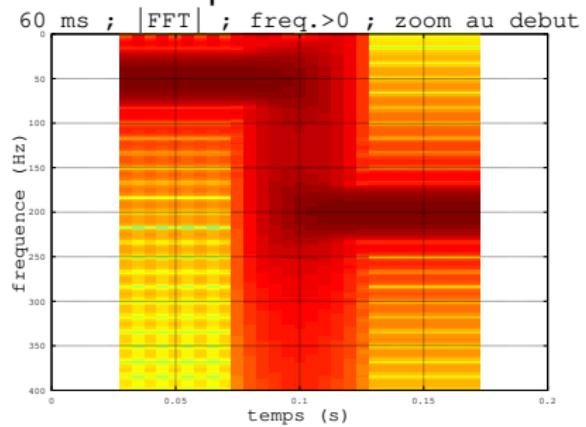
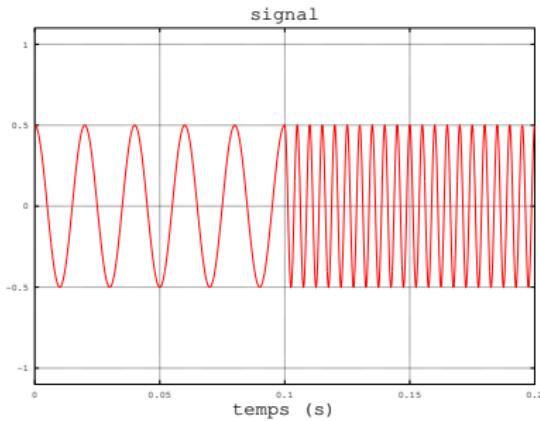


## Exemple 1 – Influence taille fenêtre

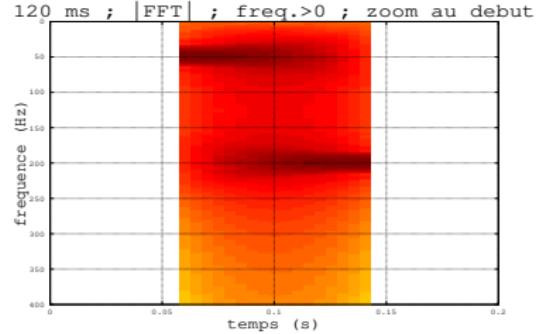
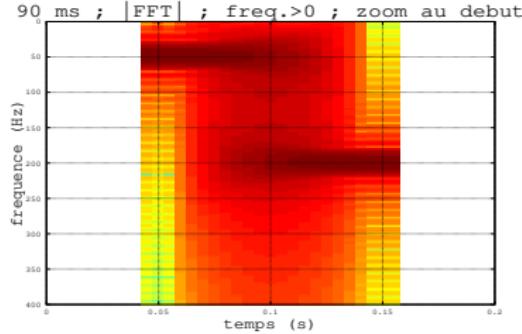
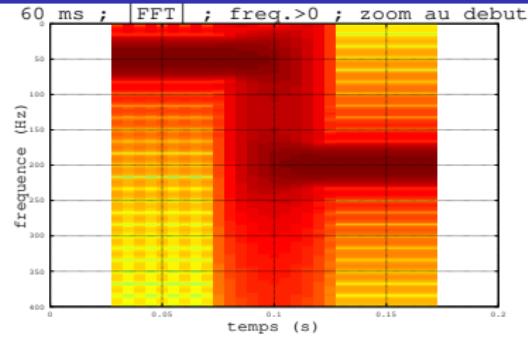
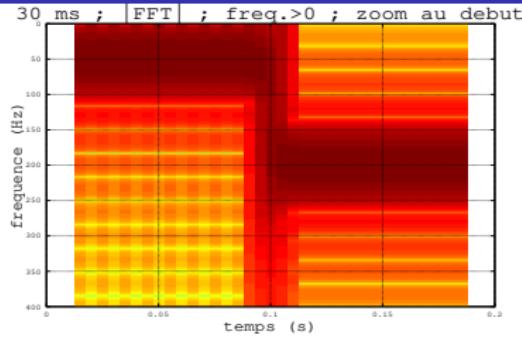


## Retour à l'exemple 2

- ▶ 2 cosinus présents successivement :  $x(t) = 0.5 \cos(2\pi 50t)$  pour  $t \in [0, 0.1]$ ;  $x(t) = 0.5 \cos(2\pi 200t)$  pour  $t \in [0.1, 0.2]$
- ▶  $h(\theta)$  est une fenêtre de Hanning large de 60 ms
- ▶ lecture de la TFCT : la couleur donne l'amplitude de la TFCT

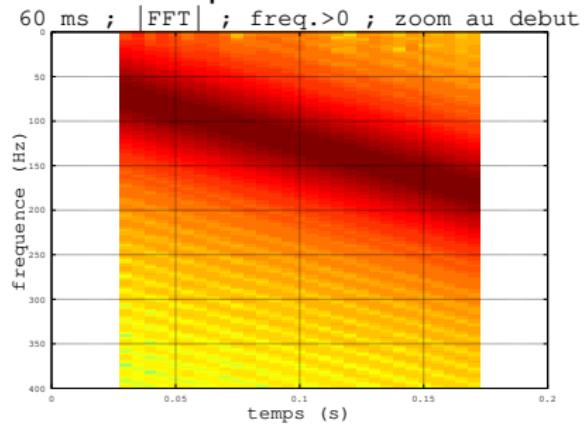
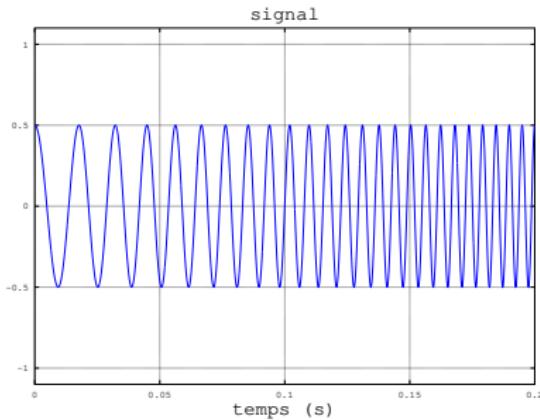


## Exemple 2 – Influence taille fenêtre

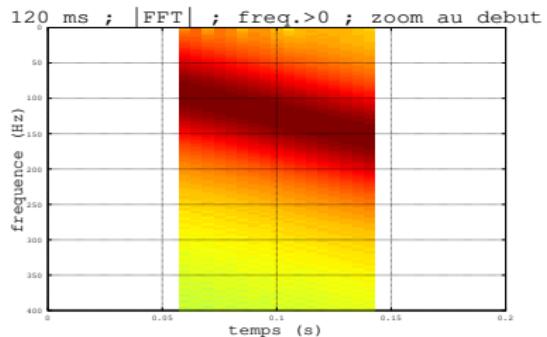
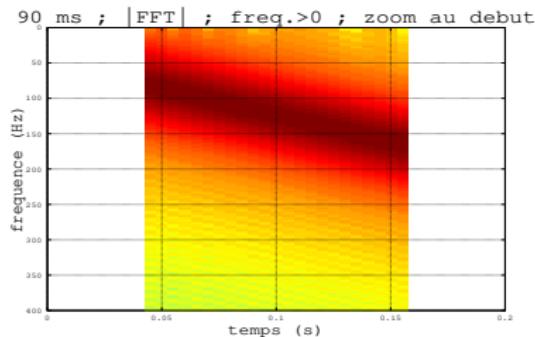
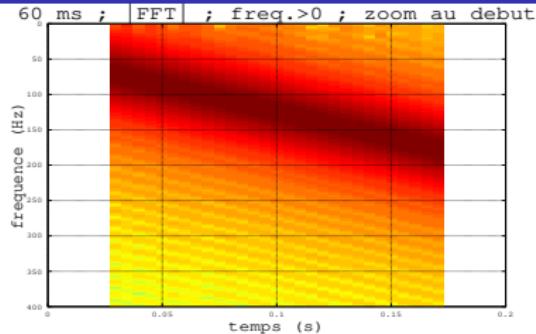
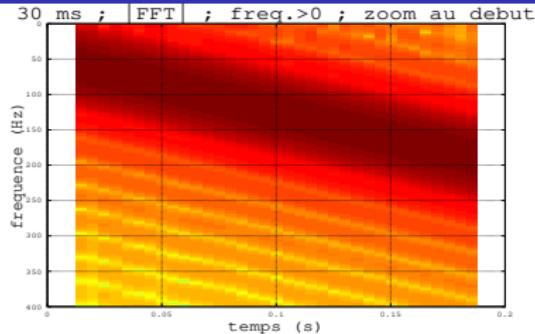


## Retour à l'exemple 4

- ▶ 1 cosinus de fréquence variant linéairement de 50 à 200 Hz :  
 $x(t) = 0.5 \cos(\phi(t))$  pour  $t \in [0, 0.2]$
- ▶  $h(\theta)$  est une fenêtre de Hanning large de 60 ms
- ▶ lecture de la TFCT : la couleur donne l'amplitude de la TFCT

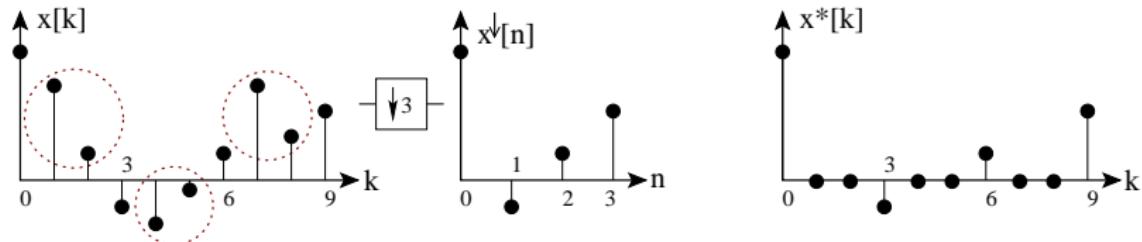


## Exemple 4 – Influence taille fenêtre

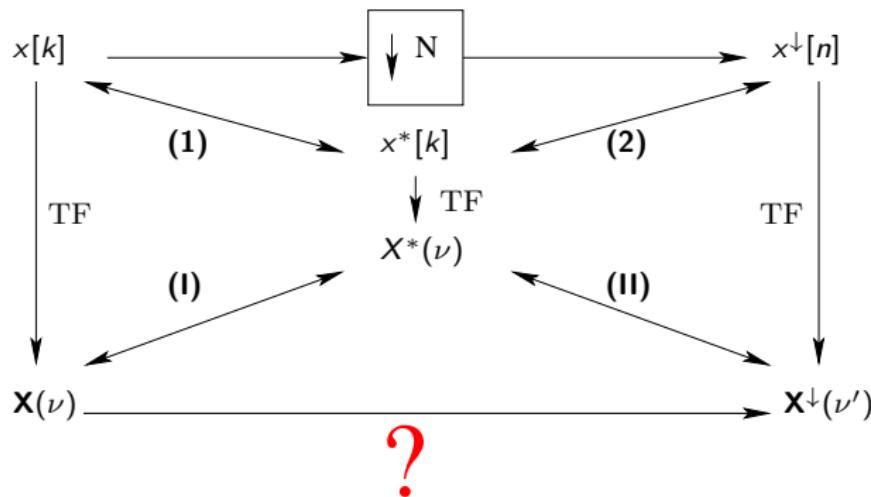


## Qu'est-ce que le sous-échantillonnage ?

- ▶ La décimation d'un facteur entier  $N$  consiste à ne garder qu'un échantillon de la suite  $x[k]$  sur  $N$
- ▶ Pour faciliter l'étude de ce système, il faut introduire le signal fictif :  $x^*[k] = x[k] \sum_n \delta[k - nN] = \sum_n x[nN] \delta[k - nN]$
- ▶ Schématiquement, on a :



## Le petit graphique habituel



(Note :  $x[nN] = x^{\downarrow}[n]$ )

## Formellement (1/3) – considérons (1)

$$x^*[k] = x[k] \sum_{n=-\infty}^{+\infty} \delta[k - nN] \quad (1)$$

Note : c'est un détail, mais on utilise la convolution circulaire (voir Signal), et pas la convolution linéaire :

$$X^*(\nu) = X(\nu) \otimes \frac{1}{N} \sum_{i=-\infty}^{+\infty} \delta\left(\nu - \frac{i}{N}\right)$$

Propriété (voir Signal) de la convolution circulaire (***N – 1 et pas +∞***) :

$$X^*(\nu) = \frac{1}{N} \sum_{i=0}^{N-1} X\left(\nu - \frac{i}{N}\right) \quad (a)$$

## Formellement (2/3) – considérons (2)

$$x^*[k] = \sum_{n=-\infty}^{+\infty} x[nN] \delta[k - nN] \quad (2)$$

Interversion des 2  $\sum$  (pour l'échantillonnage, nous avions l'interversion similaire de  $\sum$  et  $\int$ )

$$X^*(\nu) = \sum_{n=-\infty}^{+\infty} x^\downarrow[n] \exp(-j2\pi n\nu N) \quad (b)$$

(chaque  $\exp$  est la TF d'un des Diracs, la TF se faisant sur les  $k$ )

Or :

$$X^\downarrow(\nu') = \sum_{n=-\infty}^{+\infty} x^\downarrow[n] \exp(-j2\pi n\nu') \quad (c)$$

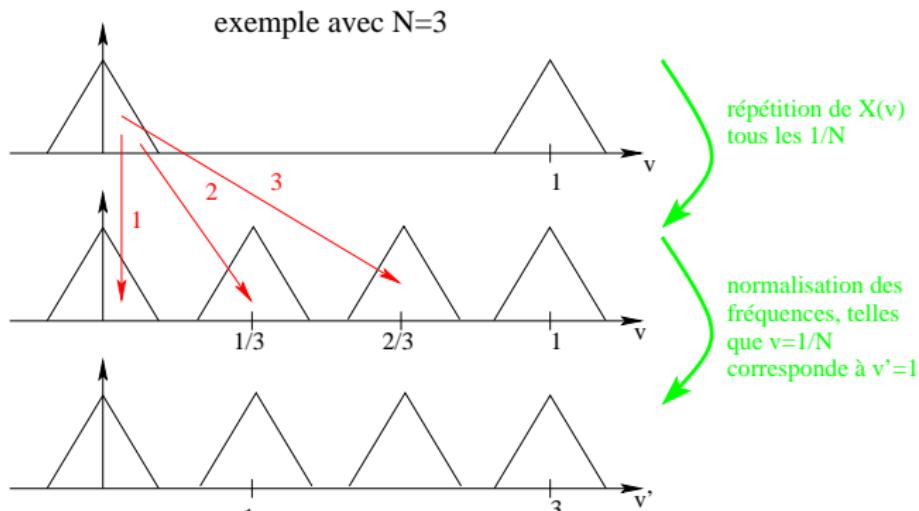
## Formellement (3/3) – lions (1) et (2)

- ▶ Donc, à l'aide de (b) et (c) on lie  $X^\downarrow(\nu')$  à  $X^*(\nu)$
- ▶ Puis à l'aide de (a) on lie  $X^\downarrow(\nu')$  à  $X(\nu)$
- ▶ Et on obtient finalement :

$$X^\downarrow(\nu') = \frac{1}{N} \sum_{n=0}^{N-1} X\left(\nu - \frac{n}{N}\right) \Big|_{\nu=\frac{\nu'}{N}}$$

## Conséquences spectrales (1/2)

Le spectre obtenu est semblable à celui qu'on aurait obtenu en échantillonnant avec  $f_e$  qui serait  $N$  fois plus faible.



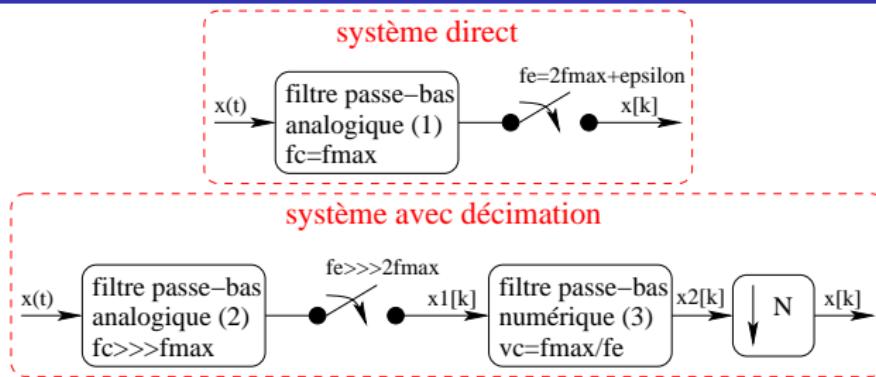
**Note : avant un décimateur, il faut placer un filtre anti-repliement (numérique)**

## Conséquences spectrales (2/2)

On aboutit au **théorème de Shannon numérique** :

- ▶ un signal à valeurs réelles et à temps discret est entièrement caractérisé par les valeurs qu'il prend tous les  $N$  points si son spectre est borné à  $\frac{1}{2N}$  sur l'intervalle  $\left[0 \quad \frac{1}{2}\right]$

## Intérêt du sous-échantillonnage

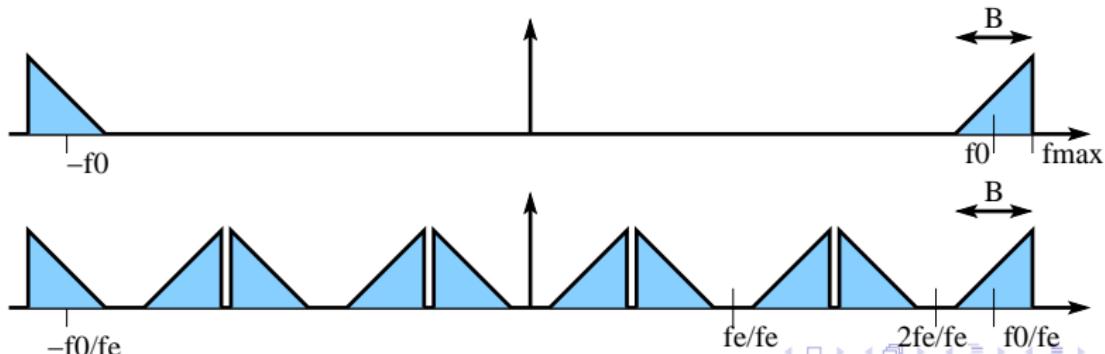


- ▶ On échantillonne avec une fréquence trop grande
  - ▶ filtre anti-repliement analogique (2) moins sélectif que (1)
  - ▶ (1) peut être difficile à réaliser
- ▶ On peut réaliser le filtre anti-repliement (3) dans le domaine numérique : c'est plus facile
- ▶ On décime pour obtenir le même nombre d'échantillons pour la suite !

## Signaux à bande étroite

Théorème de Shannon généralisé (voir Signal) :

- ▶ il faut  $f_e > 2B$  (en discret, il faut que  $2B/f_e$  soit inférieur à 1, la période du spectre)
- ▶ et  $f_e = \frac{4f_0}{2K + 1}$  (il faut aussi être sûr de pouvoir mettre un nombre pair de motifs de largeur  $B$  entre  $-f_0 + B$  et  $f_0 - B$ )



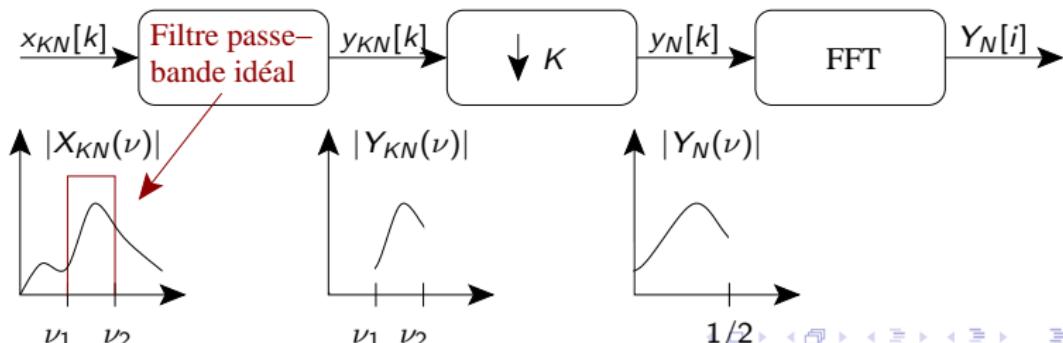
## Signaux à bande étroite

Avec le petit exemple du dessin, où on a  $f_0 = 10$  et  $B = 2$ , on obtient :

- ▶  $K = 1 \Rightarrow f_e = 13.333 > 2B = 4$  Ok!
- ▶  $K = 2 \Rightarrow f_e = 8.0000 > 2B = 4$  Ok!
- ▶  $K = 3 \Rightarrow f_e = 5.7143 > 2B = 4$  Ok!
- ▶  $K = 4 \Rightarrow f_e = 4.4444 > 2B = 4$  Ok!
- ▶  $K = 5 \Rightarrow f_e = 3.6364 < 2B = 4$  NON!

## Le zoom spectral

- ▶ On a un dispositif qui permet de calculer la FFT sur  $N$  points
- ▶ Dans le signal traité, on ne s'intéresse qu'à une petite zone fréquentielle :  $\nu \in [\nu_1, \nu_2]$
- ▶ Alors, si  $K = \frac{1}{2(\nu_2 - \nu_1)}$  est un entier (c'est facile de se ramener à cette condition, en élargissant un peu la zone), on peut utiliser la chaîne de traitement :



## Le zoom spectral

- ▶ Par exemple, si on ne s'intéresse qu'à ce qu'il y a entre  $\nu_1 = 0.25$  et  $\nu_2 = 0.5$  (la moitié du spectre), on obtient  $K = 2$  ci-dessus, et on peut décimer d'un facteur 2 (après avoir quand même utilisé un filtre passe-bande ne gardant que la bande  $\nu = [0.25 \ 0.5]$ ).
- ▶ Note : c'est ce qu'on fait pour l'analyse avec les ondelettes

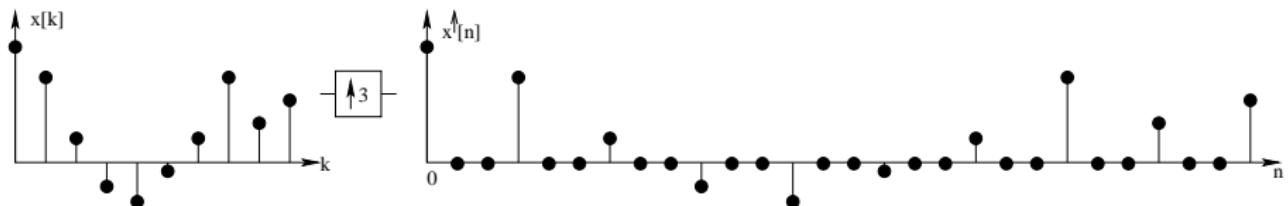
## Intérêt pour les ondelettes

Plus spécifiquement, dans ce cours, on aura besoin du sous-échantillonnage (et du sur-échantillonnage) pour les ondelettes (ainsi que de ce qu'on a vu juste ci-dessus) :

- ▶ le sous-échantillonnage, pour l'étape d'analyse
- ▶ le sur-échantillonnage, pour l'étape de reconstruction

## Qu'est-ce que le sur-échantillonnage ?

- ▶ L'élévation de cadence d'un facteur entier  $N$  consiste à ajouter  $N - 1$  zéros entre chacun des échantillons de la suite  $x[k]$
- ▶ Puis à filtrer le signal construit ainsi pour obtenir une véritable interpolation entre les échantillons de la suite  $x[k]$
- ▶ Schématiquement, on a :



## Formellement (1/3)

- ▶ Note : c'est plus simple que pour la réduction de cadence !

$$X^\uparrow(\nu') = \sum_{n=-\infty}^{+\infty} x^\uparrow[n] \exp(-j2\pi n\nu') \quad \text{et} \quad X(\nu) = \sum_{k=-\infty}^{+\infty} x[k] \exp(-j2\pi k\nu)$$

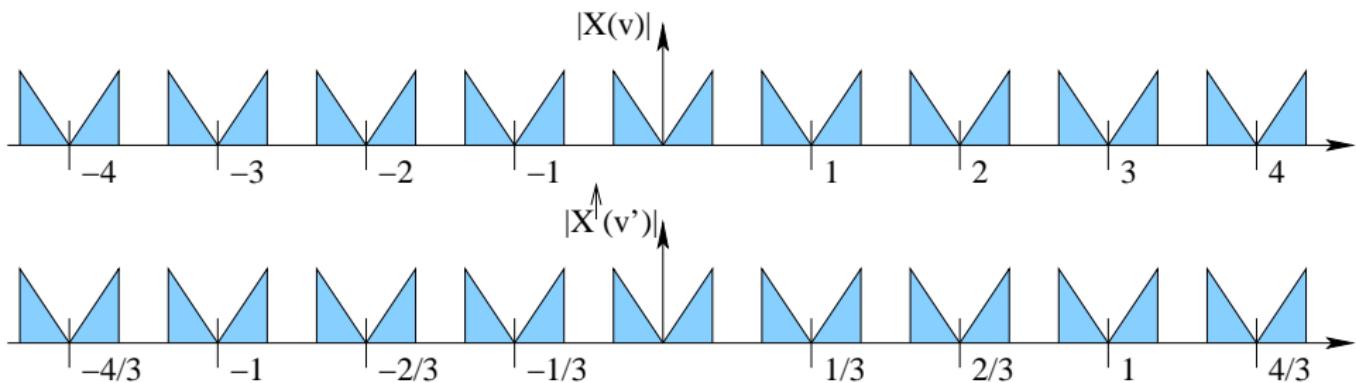
$$\text{or } x^\uparrow[n] = x[k]|_{n=kN} \quad \text{et} \quad x^\uparrow[n] = 0 \text{ pour } n \neq kN$$

$$\text{donc : } X^\uparrow(\nu') = \sum_{k=-\infty}^{+\infty} x^\uparrow[kN] \exp(-j2\pi kN\nu') \Rightarrow \nu = N\nu'$$

$$\text{finalement : } X^\uparrow(\nu') = X(\nu)|_{\nu=N\nu'}$$

## Formellement (2/3)

Exemple avec N=3



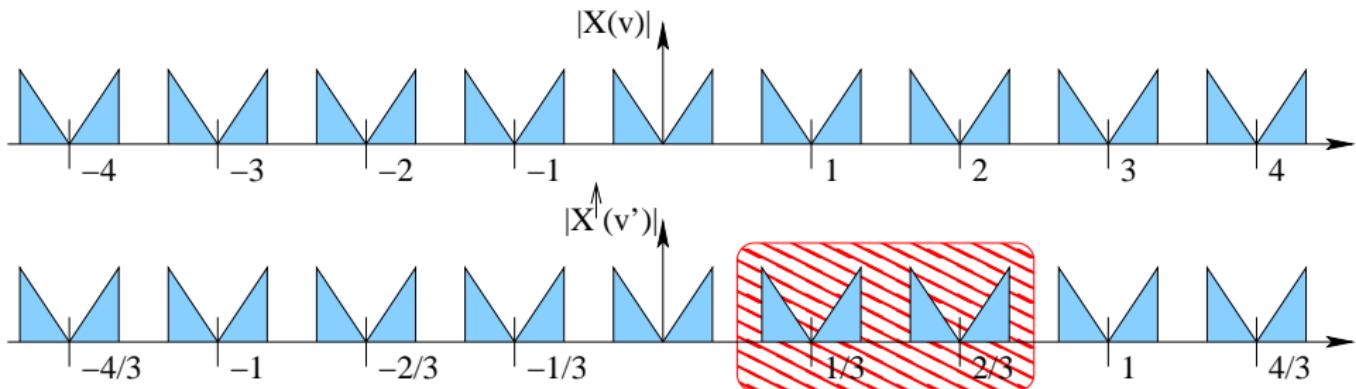
## Formellement (3/3)

Un seul effet sur le spectre (contre trois pour la réduction de cadence) :

- ▶ renormalisation de l'axe des fréquences

## Conséquences spectrales (1/2)

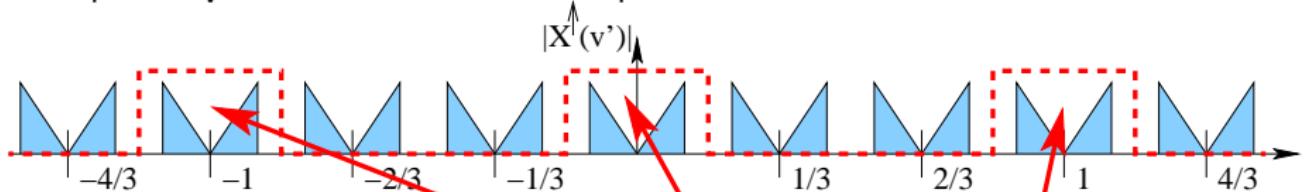
Exemple avec  $N=3$



entre les fréquences réduites  $[0 \ 1]$   
on a introduit des hautes fréquences,  
plutôt très ennuyeuses

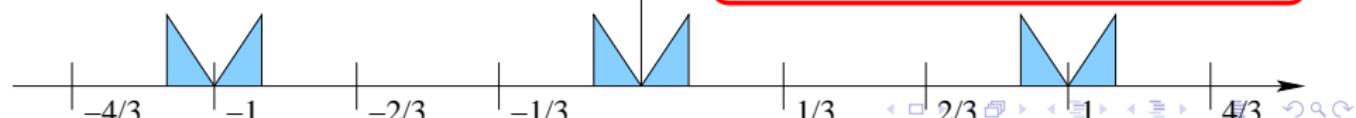
## Conséquences spectrales (2/2)

Note : pour la réduction de cadence, il y a possibilité de repliement spectral, et il faut donc filtrer **avant** d'effectuer la réduction de cadence ; par contre, pour l'élévation de cadence, il n'y a pas ce risque de repliement spectral, et c'est plutôt **après** l'élévation de cadence qu'il faut filtrer

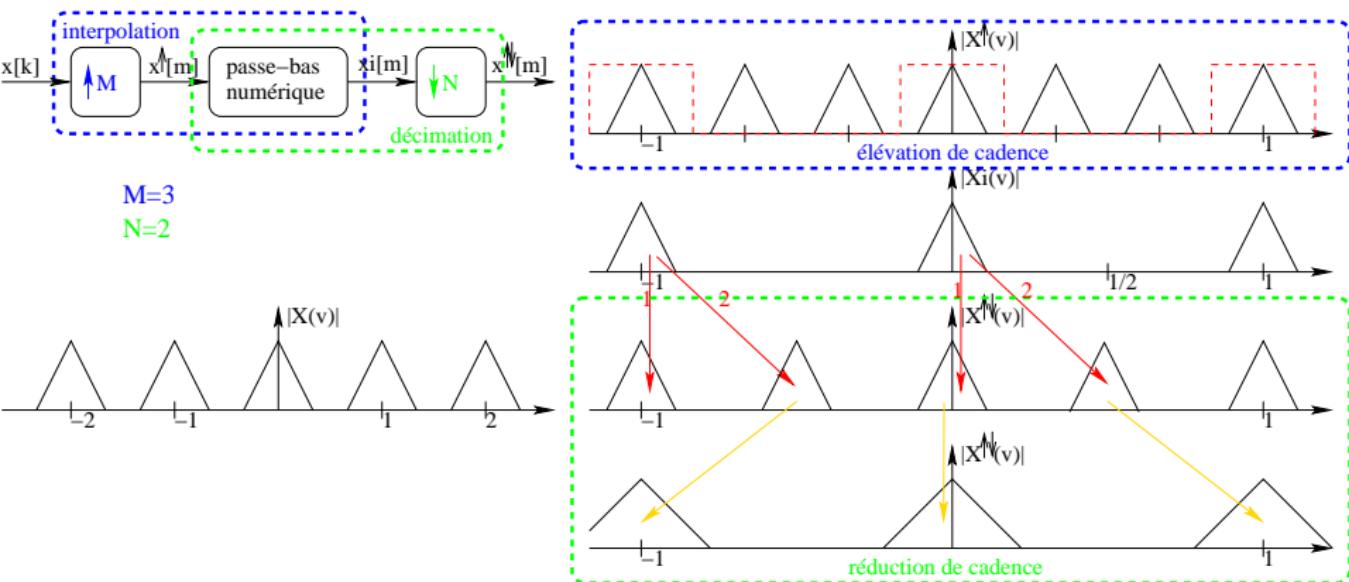


Exemple avec  $N=3$

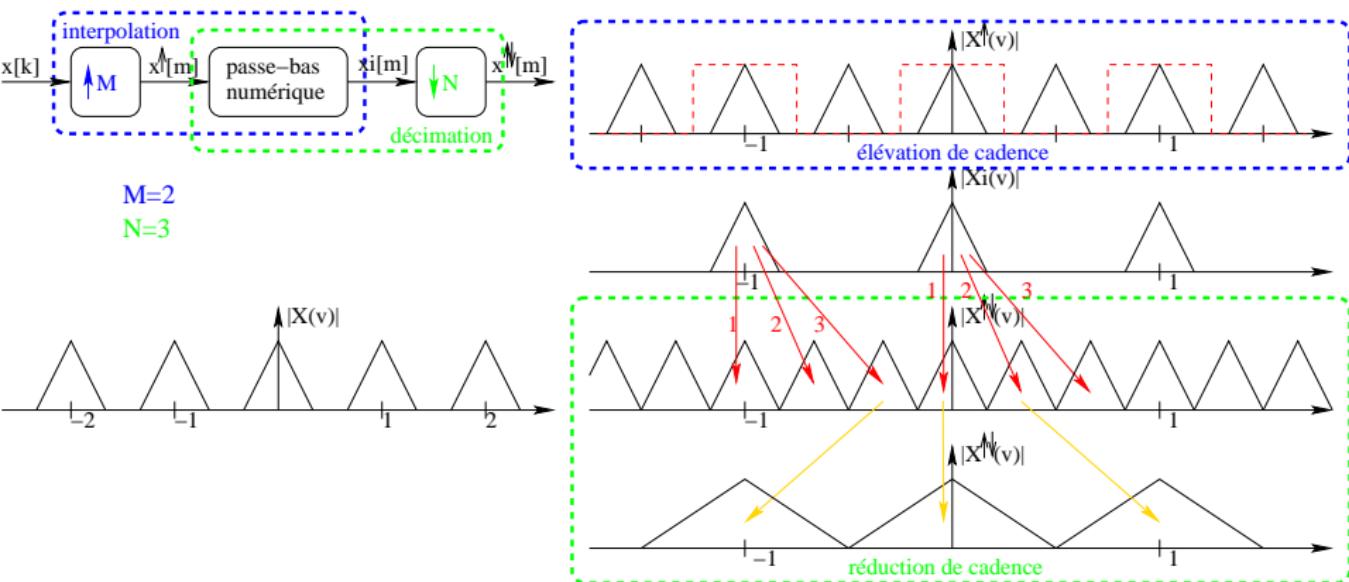
filtrage passe-bas numérique  
 idéal, ou interpolation  
 (note : bien sûr, la réponse en fréquence d'un  
 filtre numérique est elle aussi périodisée !)



## Modification de cadence d'un facteur rationnel $M/N$ (1/2)



## Modification de cadence d'un facteur rationnel $M/N$ (2/2)



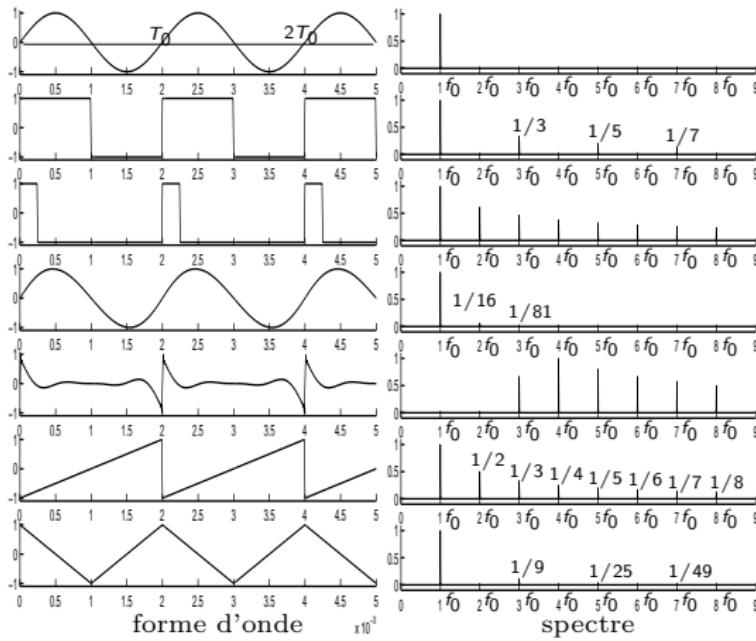
## Qu'est-ce qu'un signal harmonique ?

C'est un signal périodique. Il s'écrit sous cette forme :

$$e(t) = \sum_{i=1}^m A_i \cos(2\pi f_0 i t + \phi_i)$$

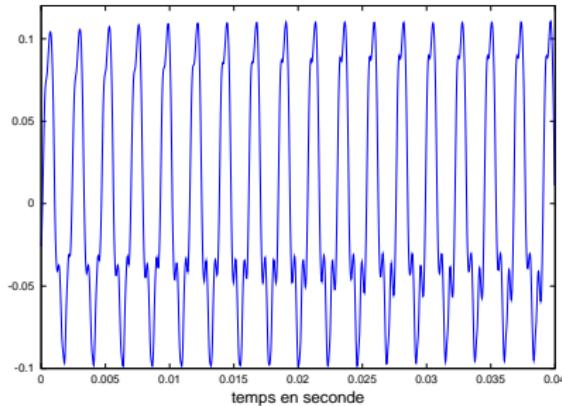
C'est-à-dire que c'est une somme de sinus dont les fréquences sont des multiples de la fréquence fondamentale  $f_0$  (et d'amplitudes quelconques)

## Exemples (venant de la musique)



## Exercice – question 1

- ▶ Écouter le son *noteflute.wav* (une note de flûte traversière)
- ▶ Une trame de ce son est représentée sur la figure :



- ▶ Que peut-on dire sur le spectre qu'on va obtenir par transformée de Fourier ?

## Exercice – solution 1

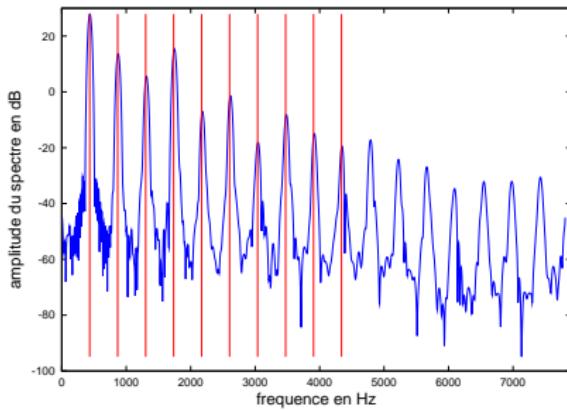
*ATTENTION : les solutions sont données sur les transparents suivants*

## Exercice – solution 1

- ▶ Le signal est (quasi-)périodique, donc on va obtenir des « raies » régulièrement espacées (souvenez-vous des séries de Fourier de signaux harmoniques)
- ▶ En fait, à cause de la troncature, on obtient des lobes larges de l'ordre de  $1/0.04 = 25 \text{ Hz}$
- ▶ Ces lobes étant séparés d'environ  $434 \text{ Hz}$  (la période du signal, lire sur la figure, étant d'environ  $2.3 \text{ ms}$ , puisqu'il y a un peu plus de 17 périodes sur  $40 \text{ ms}$ )
- ▶ Pour le reste, **on ne peut rien dire de plus sur les amplitudes et les phases respectives de chacune des composantes fréquentielles du signal**

## Exercice – solution 1

- Le spectre obtenu est donné sur la figure :



note : en rouge (segments verticaux), on a les positions des 10 premiers harmoniques (de 434 Hz à 4340 Hz)

## Signal analytique – Spectre

- ▶ Un signal analytique est un signal à valeurs complexes dont le spectre est nul pour les fréquences négatives
- ▶ Exemple :  $x(t) = \cos(2\pi f_0 t)$  n'est bien sûr pas un signal analytique  $\Rightarrow$  le signal analytique qui lui correspond est  $z(t) = \exp(j2\pi f_0 t)$
- ▶ Le filtrage de Hilbert permet, à partir d'un signal réel  $x(t)$ , de reconstruire le signal  $\sigma(t)$ , qui est la partie imaginaire du signal analytique correspondant à  $x(t)$
- ▶ Le filtrage de Hilbert suppose idéalement que  $x(t)$  est à support temporel infini, ce qui n'est jamais le cas, et donc  $\sigma(t)$  n'est jamais parfait (la réalisation pratique du filtre n'est pas forcément aisée)

## Spectre d'un signal analytique

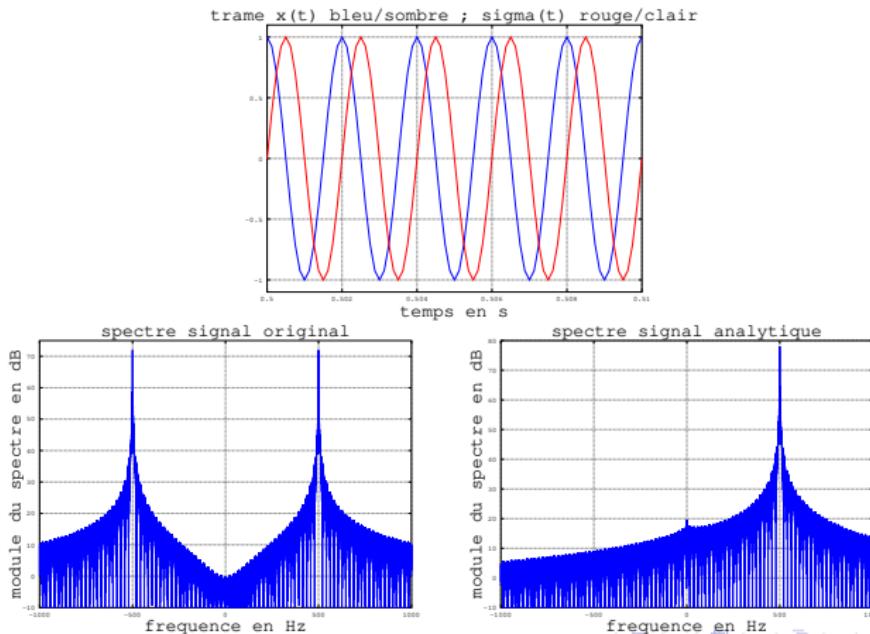
- ▶ Attention : le signal analytique associé à  $x(t) = a(t) \cos(\phi(t))$  n'est pas généralement  $z(t) = a(t) \exp(j\phi(t))$
- ▶ Le filtrage de Hilbert permet d'introduire la notion de fréquence instantanée, utile pour l'étude de signaux dont le contenu fréquentiel n'est pas très étendu
- ▶ La définition de la fréquence instantanée est :

$$f = \frac{1}{2\pi} \frac{d\phi(t)}{dt}$$

- ▶ Remarque : si on a efficacement accès au signal analytique, il n'y a pas lieu alors de calculer de TF

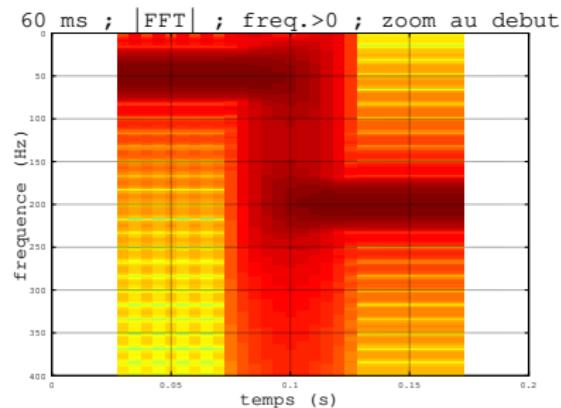
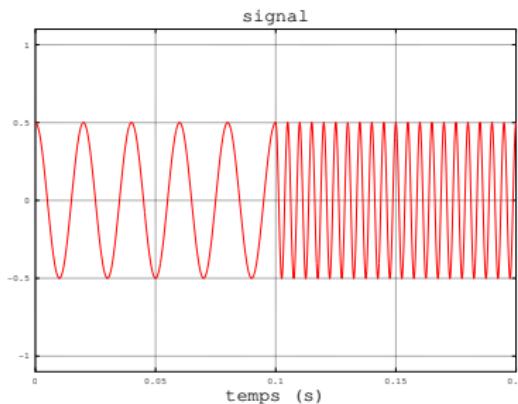
## Spectre d'un signal analytique

- ▶ L'exemple précédent, en pratique (voir `exe_hilbert.m`) :



## Hilbert – Exemple 2 – question 1

- ▶ On prend ce signal (qu'on reverra) :
  - ▶ 2 cosinus présents successivement
  - ▶  $x(t) = 0.5 \cos(2\pi 50t)$  pour  $t \in [0, 0.1]$  ;  $x(t) = 0.5 \cos(2\pi 200t)$  pour  $t \in [0.1, 0.2]$



- ▶ Est-il légitime d'utiliser le signal analytique dans ce cas ? Pourquoi ?

## Exemple 2 – solution 1

- ▶ Pour pouvoir utiliser la méthode du signal analytique, plutôt que le spectrogramme (ou en concurrence avec lui), il faut que le signal soit de contenu fréquentiel peu étendu
- ▶ C'est le cas, sauf au moment de la transition
- ▶ Au moment de la transition, on s'attend donc à ce qu'on obtienne des résultats aberrants ⇒ de toute façon ceux obtenus grâce au spectrogramme risquent de ne pas être fabuleux, vu la tête du signal

## Exemple 2 – question 2

- ▶ Pour calculer le signal analytique, on utilise la fonction de matlab/python toute faite : **hilbert/scipy.signal.hilbert**
  - ▶ avant de programmer quelque chose, vérifiez toujours si ça n'existe pas déjà dans matlab/octave/python
  - ▶ et, si vous ne savez pas comment utiliser la chose, n'oubliez pas l'aide en ligne de matlab/octave/python
- ▶ Pour trouver la fréquence instantanée, on utilise la dérivée numérique. C'est-à-dire ?
- ▶ Comment trouve-t'on le trajet de la fréquence grâce au spectrogramme ?
- ▶ Mettre en regard la fréquence trouvée grâce au spectrogramme et la fréquence trouvée grâce au signal analytique

## Exemple 2 – solution 2

- ▶ Voir `nstat_exe2_tfcthil.m`
- ▶ La dérivée numérique est (ordre 1) :

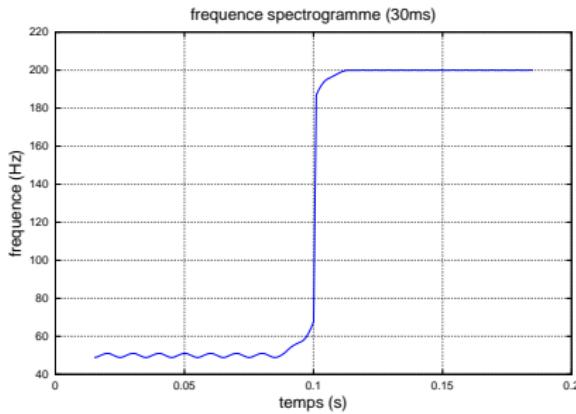
$$f = \frac{1}{2\pi} (\phi[k+1] - \phi[k]) f_e$$

C'est-à-dire qu'on a besoin de seulement 2 échantillons pour estimer la fréquence, alors qu'avec le spectrogramme on a besoin d'une trame de quelques millisecondes

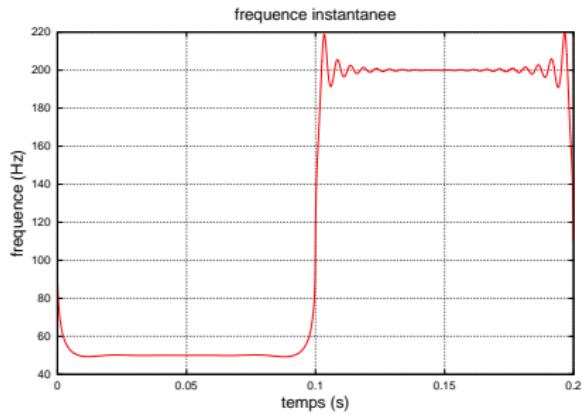
- ▶ En ce qui concerne le spectrogramme, il suffit de détecter le maximum de chaque spectre, et la position de chacun de ces maximums (note : la fonction **max** de matlab renvoie la valeur du maximum et sa position ; lire l'aide en ligne de matlab pour apprendre à s'en servir)

## Exemple 2 – solution 2 – résultats

Résultat spectrogramme :



Résultat fréquence instantanée :



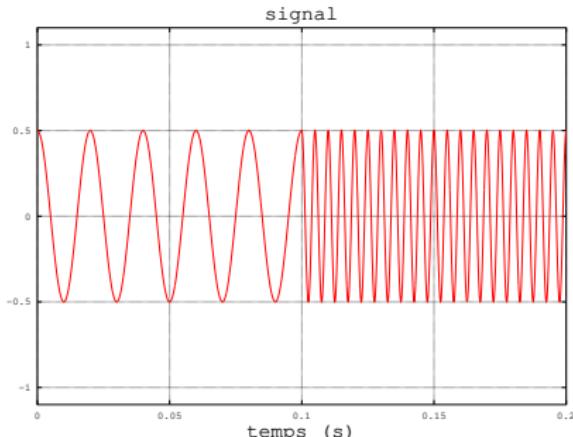
## Exemple 2 – question 3

- ▶ La fréquence instantanée semble miraculeuse : le principe d'incertitude de Gabor semble complètement dépassé, puisque deux échantillons temporels suffisent pour estimer la fréquence du signal
- ▶ Bien sûr, ce n'est pas le cas : les miracles n'existent pas
- ▶ Qu'est-ce qui se passe, alors ?

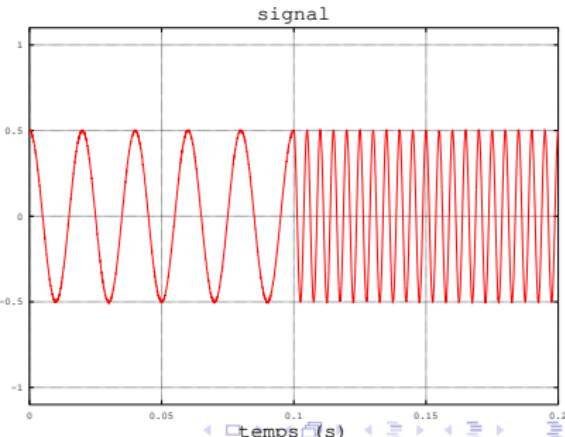
## Exemple 2 – solution 3 (1/4)

- ▶ en fait, l'exemple tel que présenté est un cas d'école, qui n'existe jamais dans la réalité : il n'y a pas de bruit
- ▶ dès qu'on ajoute un tout petit peu de bruit, la fréquence instantanée s'effondre (voir nstat\_exe2\_tfcthilbruit.m)

signal sans bruit :

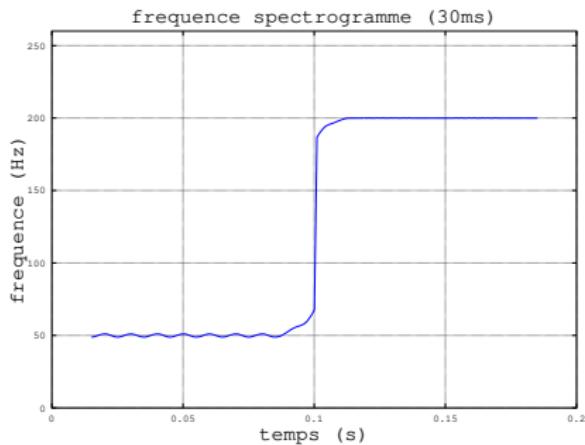


signal avec bruit :

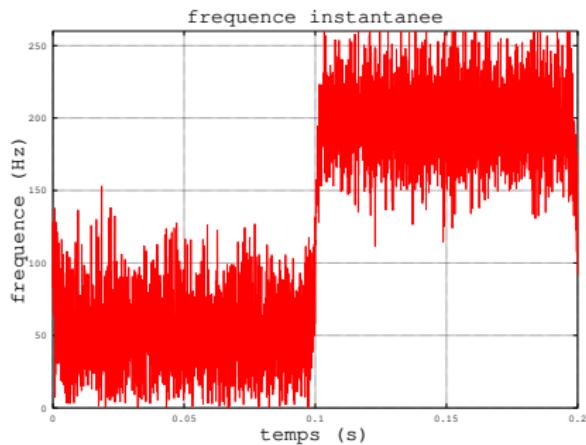


## Exemple 2 – solution 3 (2/4)

résultat spectrogramme :



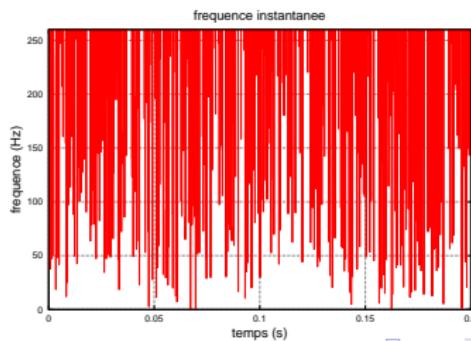
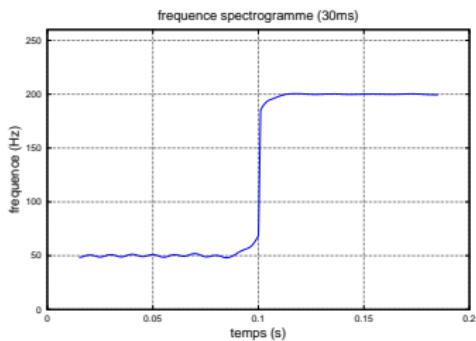
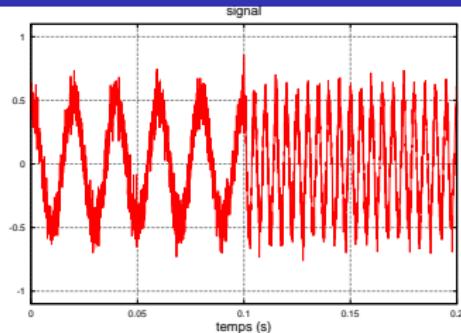
résultat fréquence instantanée :



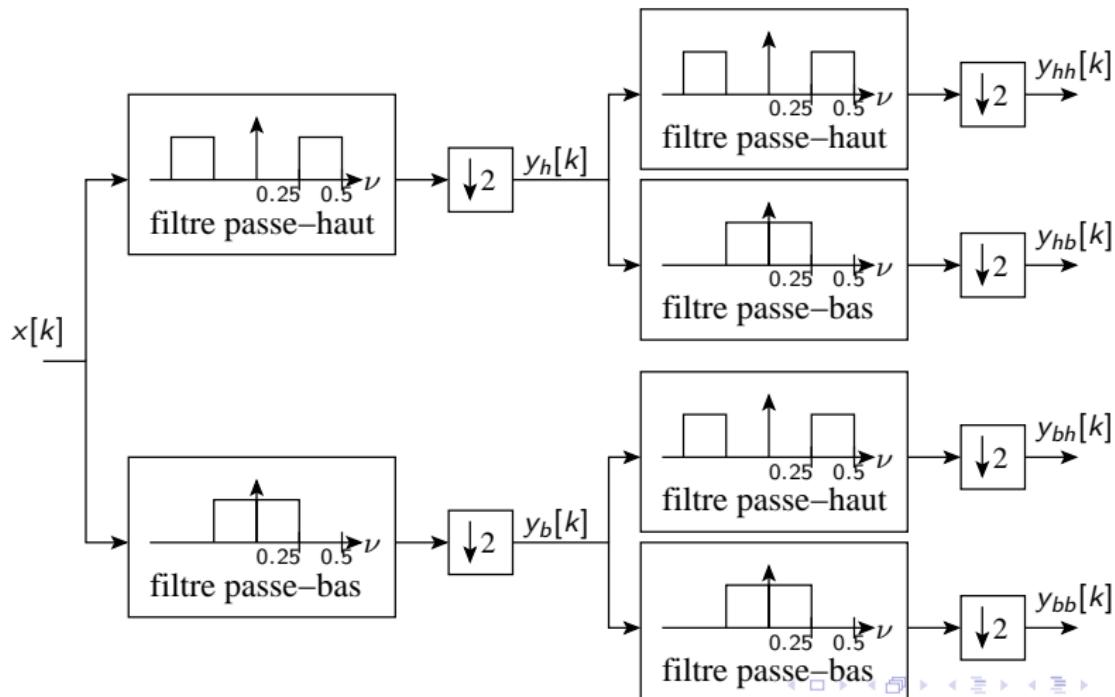
## Exemple 2 – solution 3 (3/4)

- ▶ Pour s'en sortir avec la fréquence instantanée, on pourrait imaginer de lisser la courbe obtenue, sur des largeurs de trames de quelques millisecondes... et alors le principe d'incertitude de Gabor reviendrait au premier plan
- ▶ Le bruit ci-dessus était de variance très petite :  $\sigma^2 = 1.6 \cdot 10^{-5}$ , ce qui donne un RSB de 39 dB (autant dire qu'il n'y a pas de bruit)
- ▶ Sur le transparent suivant, on a ce qu'on obtient quand le RSB est de 11 dB : on voit que même en lissoant, il n'est pas évident qu'on arrive à quelque chose (voir `nstat.exe2_tfcthilbruit2.m`)

## Exemple 2 – solution 3 (4/4)

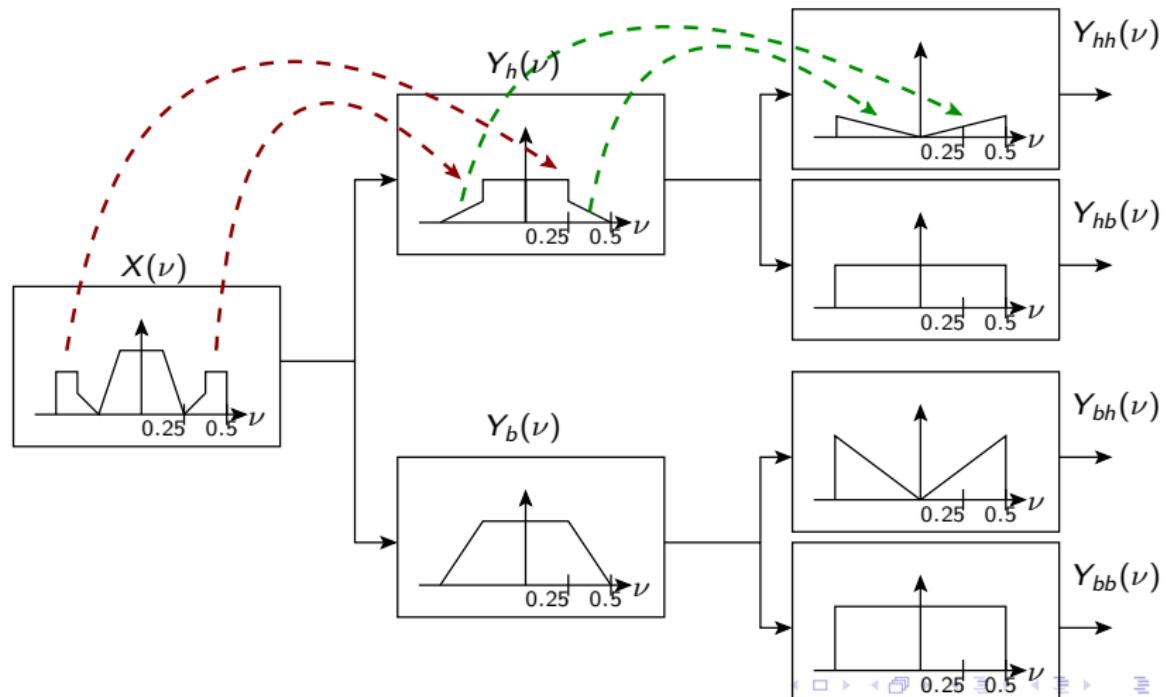


## Une cascade de filtres passe-bande



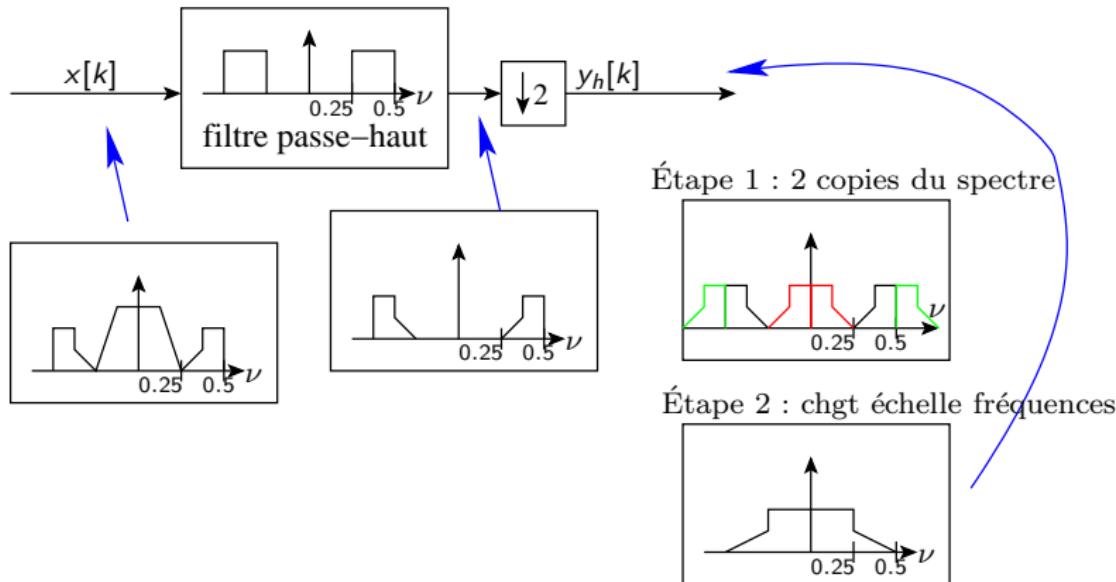
et on continue le processus

## Et les spectres des signaux obtenus



et on continue le processus

## Un exemple pour les filtres passe-haut



## Et un rappel important (au cas où)

Souvenez-vous des SLI (Systèmes Linéaires et Invariants) :

- ▶ Nos filtres ici sont bien sûr des SLI
- ▶ Et donc, leur sortie est égale au produit de convolution entre leur entrée et leur réponse impulsionnelle
- ▶ C'est-à-dire :
  - ▶ en discret :  $s[k] = \sum_n e[n]h[k - n]$
  - ▶ en continu :  $s(t) = \int e(\theta)h(t - \theta)d\theta$

## Un exemple

- ▶ On a un signal harmonique composé de la somme de 4 sinus. La fréquence fondamentale est 880 Hz et la fréquence d'échantillonnage est de 8000 Hz.
- ▶ Combien de décompositions successives faut-il pour récupérer un sinus au plus par bande ?
- ▶ À quelles fréquences ont été du coup transposées les 4 sinus initiaux ?
- ▶ Quelle méthode utiliser pour trouver les fréquences de ces 4 sinus ?

## Exercice – solution 1

*ATTENTION : les solutions sont données sur les transparents suivants*

## Exercice – solutions 1 et 2

- ▶ Première décomposition – Partie basses fréquences
  - ▶  $f_e$  devient  $f_{e1} = 4000$  Hz
  - ▶ Deux sinus, de fréquences inchangées : 880 Hz et 1760 Hz
- ▶ Première décomposition – Partie hautes fréquences
  - ▶  $f_e$  devient  $f_{e1} = 4000$  Hz
  - ▶ Deux sinus, de fréquences modifiées
    - ▶ bornes de la bande : ce qui était à -4000 Hz va se retrouver à 0 Hz et ce qui était à -2000 Hz va se retrouver à 2000 Hz
    - ▶ le sinus à  $-3f_0 = -2640$  Hz va se retrouver à 1360 Hz
    - ▶ le sinus à  $-4f_0 = -3520$  Hz va se retrouver à 480 Hz

## Exercice – solutions 1 et 2

- ▶ Deuxième décomposition – Partie basses fréquences des basses fréquences
  - ▶  $f_e$  devient  $f_{e2} = 2000$  Hz
  - ▶ Un sinus, de fréquence inchangée : 880 Hz
- ▶ Deuxième décomposition – Partie hautes fréquences des basses fréquences
  - ▶  $f_e$  devient  $f_{e2} = 2000$  Hz
  - ▶ Un sinus, de fréquence modifiée
    - ▶ bornes de la bande : ce qui était à -2000 Hz va se retrouver à 0 Hz et ce qui était à -1000 Hz va se retrouver à 1000 Hz
    - ▶ le sinus à  $-2f_0 = -1760$  Hz va se retrouver à 240 Hz

## Exercice – solutions 1 et 2

- ▶ Deuxième décomposition – Partie basses fréquences des hautes fréquences
  - ▶  $f_e$  devient  $f_{e2} = 2000$  Hz
  - ▶ Un sinus, de fréquence inchangée : 480 Hz (c'était initialement, le sinus à  $4f_0 = 3520$  Hz)
- ▶ Deuxième décomposition – Partie hautes fréquences des hautes fréquences
  - ▶  $f_e$  devient  $f_{e2} = 2000$  Hz
  - ▶ Un sinus, de fréquence modifiée
    - ▶ bornes de la bande : ce qui était à -2000 Hz va se retrouver à 0 Hz et ce qui était à -1000 Hz va se retrouver à 1000 Hz
    - ▶ le sinus à -1360 Hz va se retrouver à 640 Hz (c'était initialement, le sinus à  $3f_0 = 2640$  Hz)

## Exercice – solutions 1 et 2

Résumé :

- ▶ 2 niveaux de décomposition sont suffisants : on se retrouve avec 4 bandes, et un sinus par bande
- ▶ le sinus à  $f_0 = 880$  Hz se retrouve dans la première bande avec une fréquence de 880 Hz
- ▶ le sinus à  $2f_0 = 1760$  Hz se retrouve dans la deuxième bande avec une fréquence de 240 Hz
- ▶ le sinus à  $3f_0 = 2640$  Hz se retrouve dans la quatrième bande avec une fréquence de 640 Hz
- ▶ le sinus à  $4f_0 = 3520$  Hz se retrouve dans la troisième bande avec une fréquence de 480 Hz

## Exercice – solution 3

- ▶ On peut utiliser le filtrage de Hilbert, bande par bande, pour récupérer la fréquence de chaque sinus (et, bien sûr, il faut faire les transpositions adéquates pour retrouver les vraies fréquences des sinus du signal original)
- ▶ Bien sûr, dans le cas général, ce ne sera pas Hilbert qui sera utilisé : nous utiliserons les ondelettes

## Sujet de TL – Partie 1a

- ▶ 1. Reproduisez l'exercice (dans le langage que vous préférez : Matlab/Octave/Python)

# Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ **Partie 3 : Analyse multirésolution (ondelettes, etc.)**
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ Partie 7 : Conclusion

# Transformée en ondelettes continue

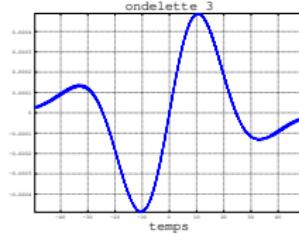
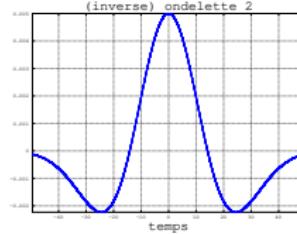
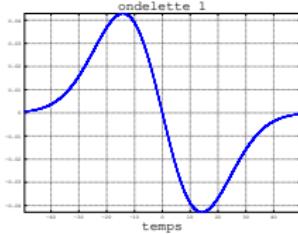
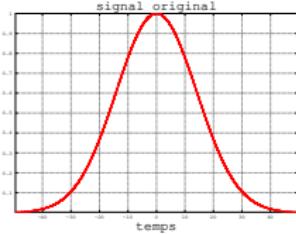
- ▶ Il s'agit d'une généralisation de l'interprétation 2 de la TFCT
    - ▶ on projette le signal sur des ondes évanescentes (autres que celles utilisées pour la TFCT : slide 46)
    - ▶ ces ondes évanescentes sont l'ondelette-mère, translatée et dilatée
- > Vient de la TFCT, l'objectif c'est d'avoir un truc qui prenne en compte les variations de phases aussi
- $$\Psi_{t,a}(\theta) = \frac{1}{\sqrt{a}} \Psi\left(\frac{\theta - t}{a}\right)$$
- ▶  $a$  est l'échelle de dilatation : on appelle cette opération une analyse multi-échelle, ou multi-résolution
  - ▶ La transformée en ondelettes est alors (à comparer à  $S_x(t, f)$ ) :

$$T_x(t, a) = \langle x, \Psi_{t,a}(\theta) \rangle = \int_{-\infty}^{+\infty} x(\theta) \frac{1}{\sqrt{a}} \Psi\left(\frac{\theta - t}{a}\right) d\theta$$

## Transformée en ondelettes continue

- ▶ Ainsi, on a, pour une échelle  $a$  donnée :
  - ▶ l'atome  $(t, f)$  centré en  $f$  (ou  $1/a$ ) est de largeur  $\Delta f$ , proportionnelle à  $f$  (ou  $1/a$ ) ; aussi, on a :  

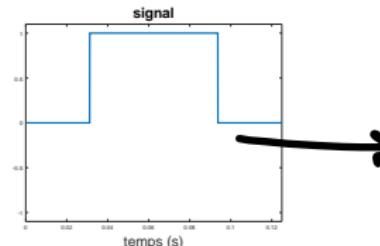
$$\Delta f \times f = \text{cste} \text{ (pour la TFCT, on a } \Delta f = \text{cste)}$$
  - ▶ ainsi : plus la fréquence d'un signal est basse, plus il faut d'échantillons temporels pour bien la détecter ; alors : la transformée en ondelettes adapte la résolution à la zone fréquentielle analysée
- ▶ Exemples d'ondelettes : les dérivées de  $h(\theta) = \exp(-\lambda\theta^2)$



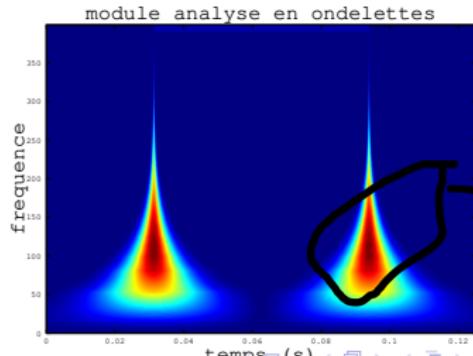
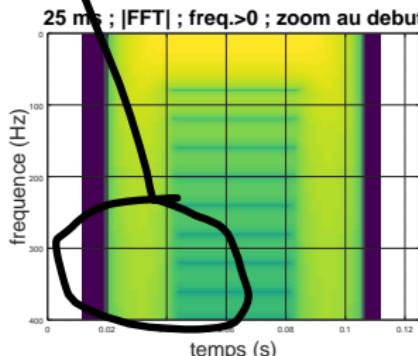
## Transformée en ondelettes continue

Soit un signal avec de forts transitoires (voir `ondelettes2a.m` et `nstat_exe5_tfct.m`)

on voit pas trop où est le transitoire



très basse fréquence donc privilégier une transformée en ondelette



là on voit bien où il y a le transitoire

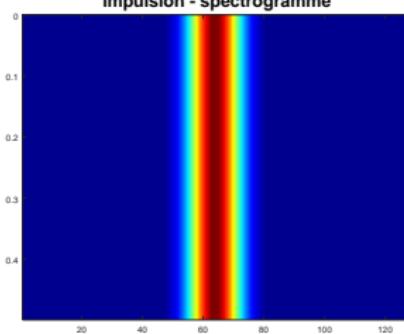
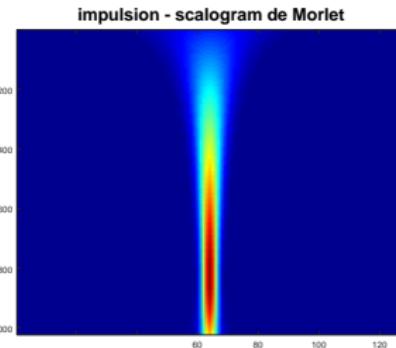
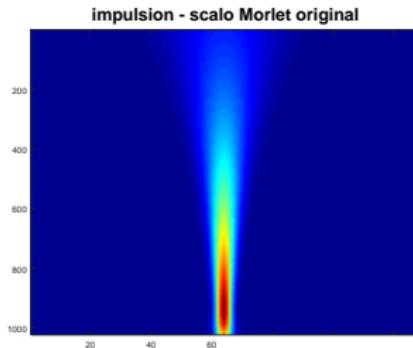
## Transformée en ondelettes continue

- ▶ En fait, les échelles  $a$ , dans les outils disponibles, ne sont pas prises continuement :
  - ▶ elles sont placées logarithmiquement
  - ▶ et comme la relation entre  $a$  et la fréquence en Hz est du type  $f = 1/a$ , si on veut des fréquences linéairement placées, il faut choisir des  $a$  en  $1/i$  où  $i$  est le numéro d'ordre de l'échelle considérée  $\Rightarrow$  j'ai modifié le code de l'outil que j'utilise ci-dessous pour obtenir ceci
  - ▶ note : en fait, pour les ondelettes discrètes, qu'on va voir plus tard, les  $a$  sont nécessairement choisies de telle façon que l'on n'a pas une analyse linéaire en fréquence

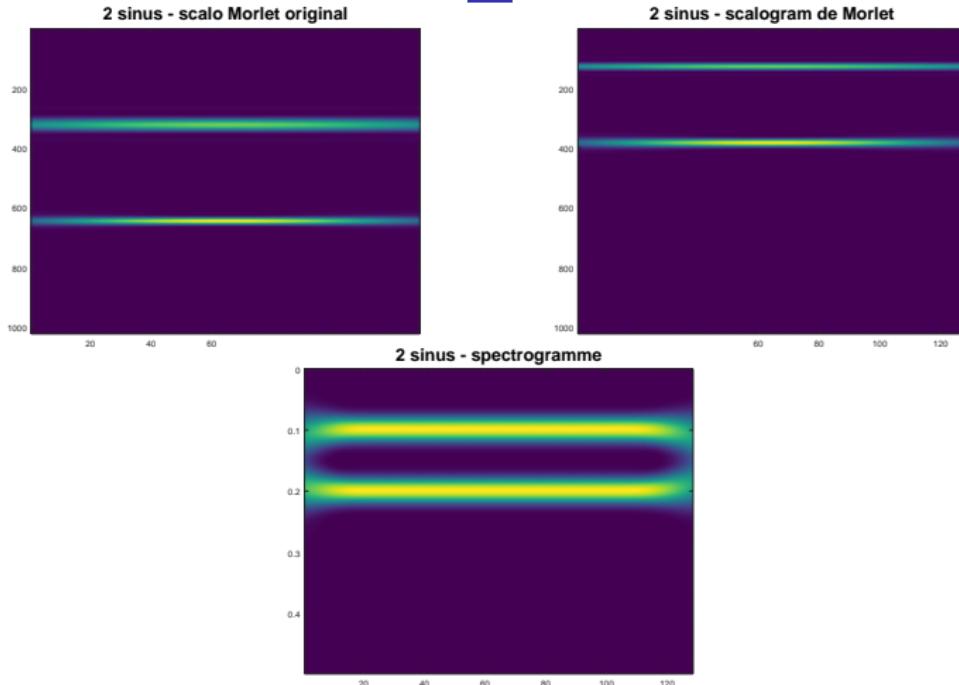
## Transformée en ondelettes continue

- ▶ Les exemples suivants ont été faits avec :
  - ▶ “lagis\_study\_scalogram.m”
  - ▶ et l’outil “tftb” modifié par moi : notamment “tfrscalo.m”
- ▶ Note : pour les scalogrammes, la position des raies dépend des bornes en fréquences prises :
  - ▶ [0.05 0.45] pour les 2 premiers exemples
  - ▶ contre [0.0025 0.4975] pour les autres

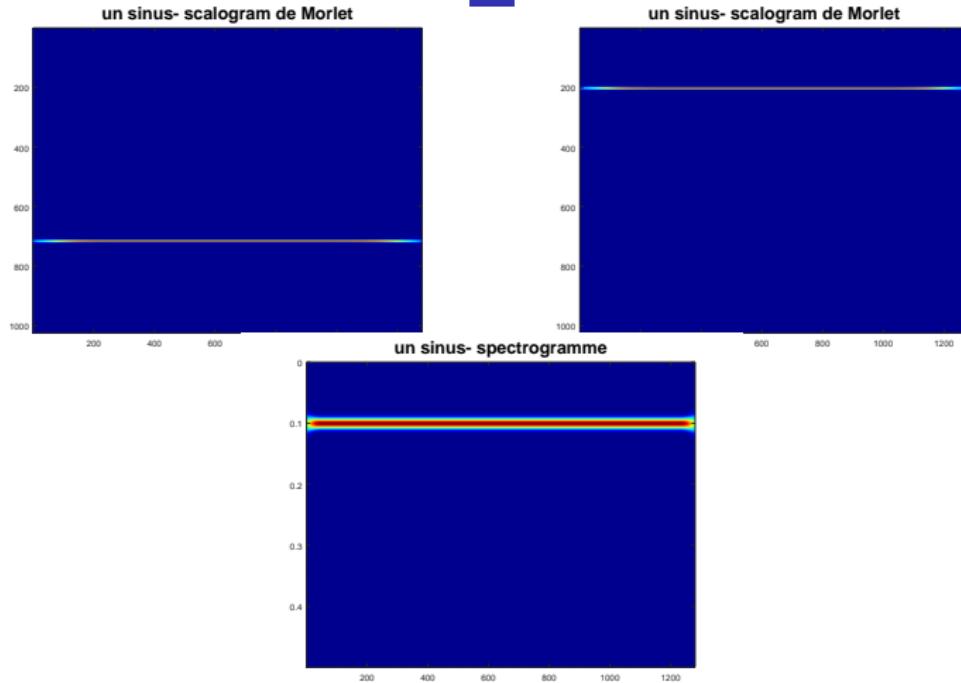
## Ondelettes continues – exemple -1 : un dirac



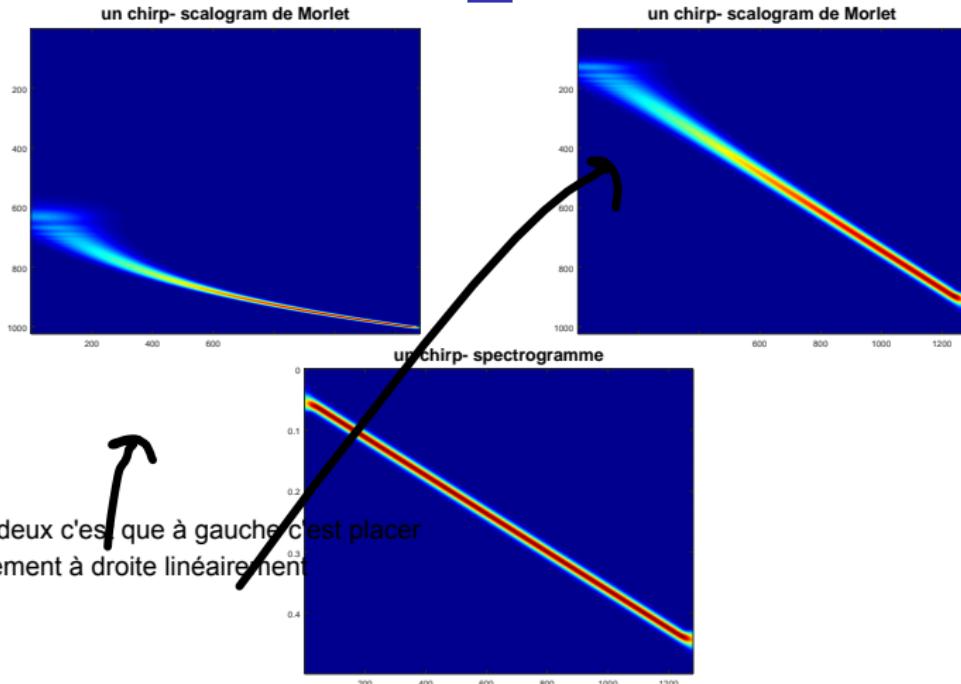
## Ondelettes continues – exemple 0 : 2 sinus



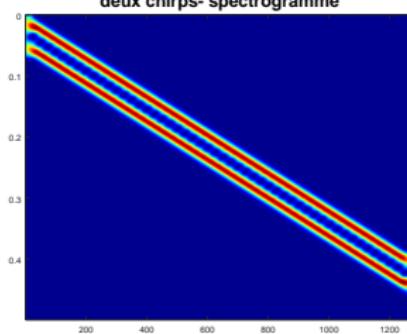
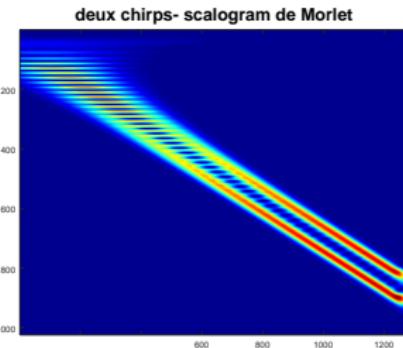
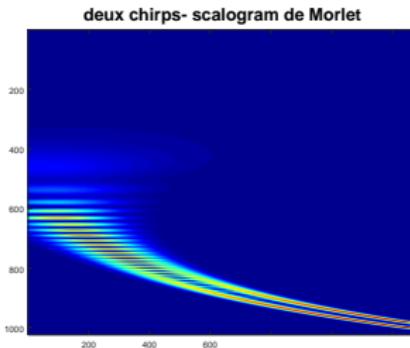
## Ondelettes continues – exemple 1 : 1 sinus



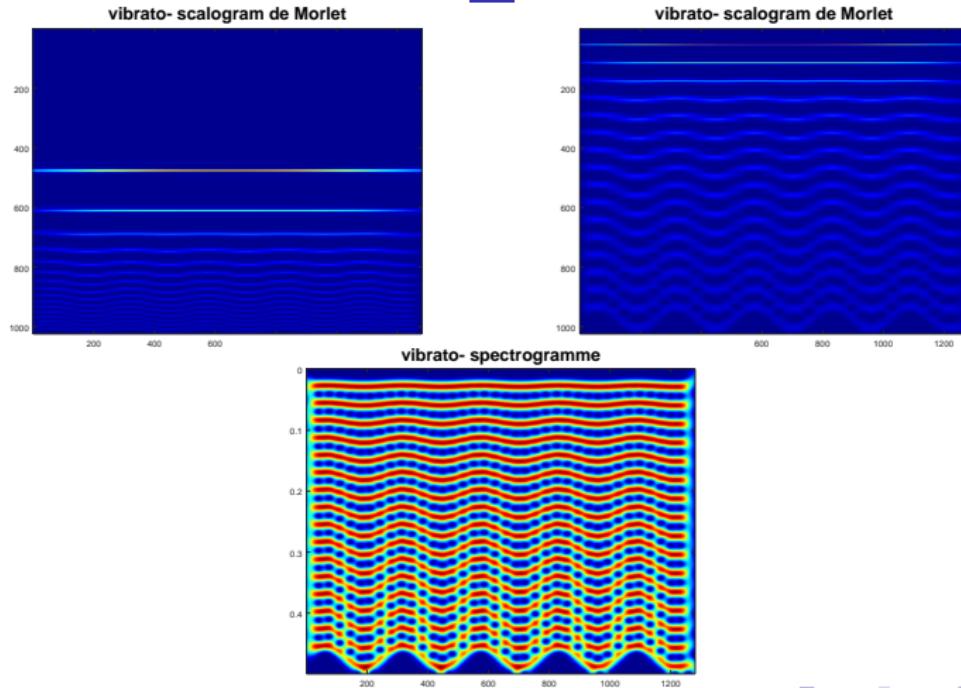
## Ondelettes continues – exemple 2 : chirp en fréquence



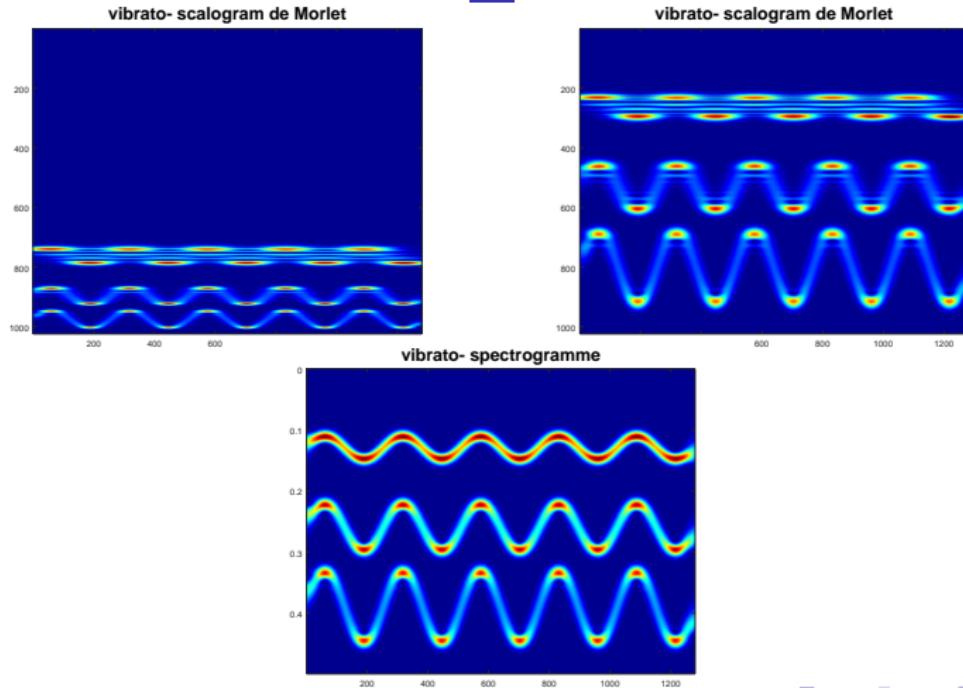
## Ondelettes continues – exemple 3 : 2 chirps en fréquence



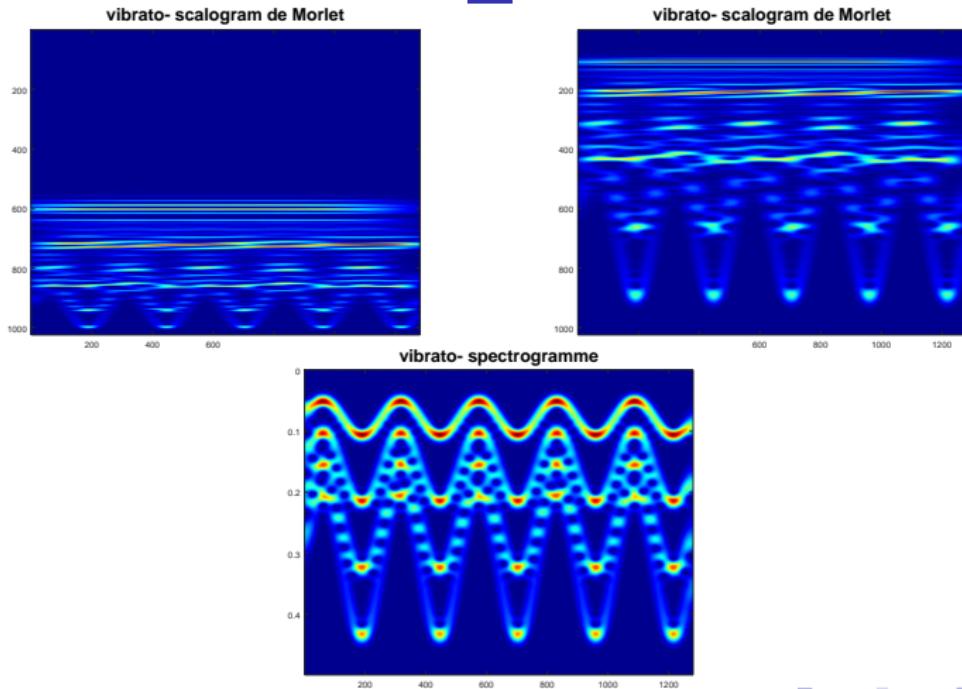
## On° continues – ex. 4 : sig harmo modulé en fréquence



## On° continues – ex. 5 : sig harmo modulé en fréquence



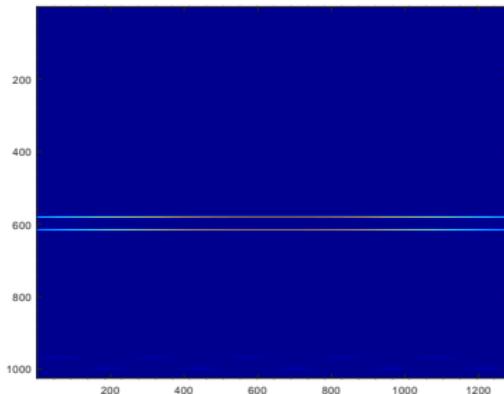
## On<sup>o</sup> continues – ex. 8 : sig harmo très modulé en fréquence



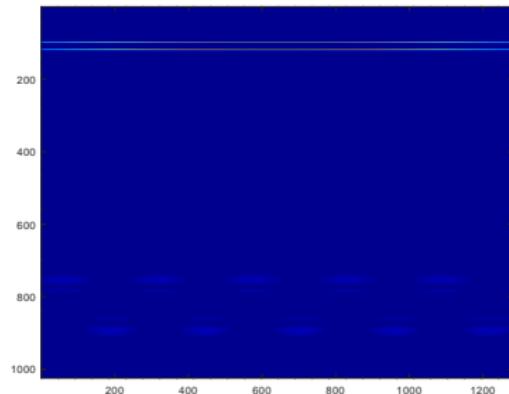
# On<sup>o</sup> continues – ex. 6 : 2 sinus proches et 1 sinus modulé

Scalogrammes :

2 sin proches+1 sin module- scalogram de Morlet

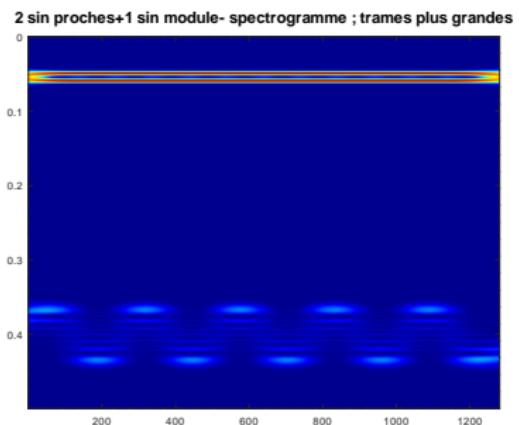
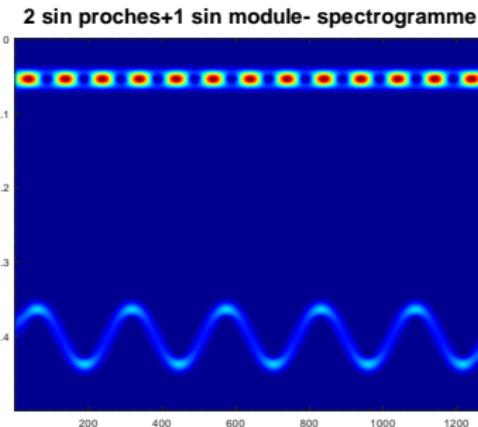


2 sin proches+1 sin module- scalogram de Morlet



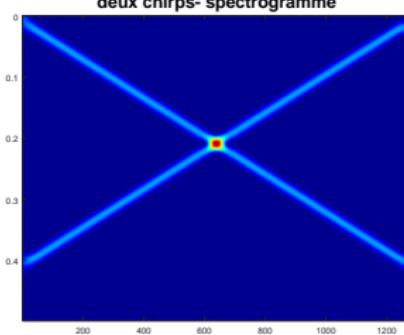
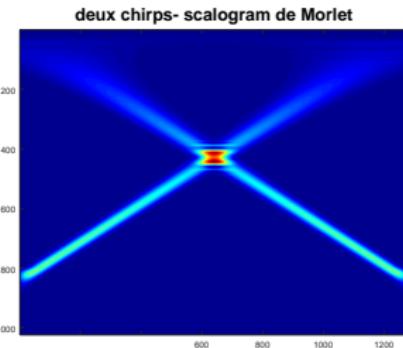
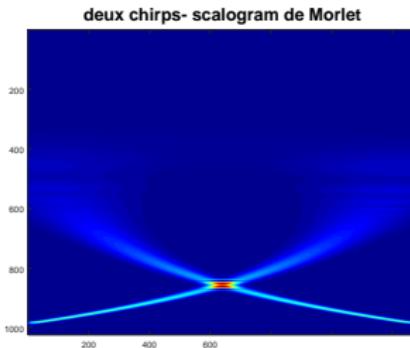
## On<sup>o</sup> continues – ex. 6 : 2 sinus proches et 1 sinus modulé

Spectrogrammes :



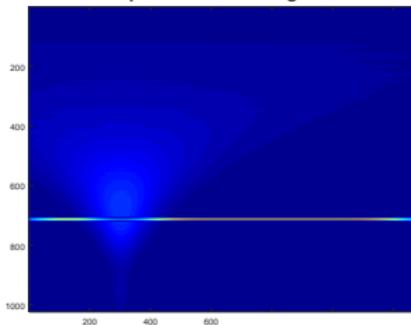
⇒ Regarder aussi avec l'autre colormap

## Ondelettes continues – exemple 7 : 2 chirps se croisant

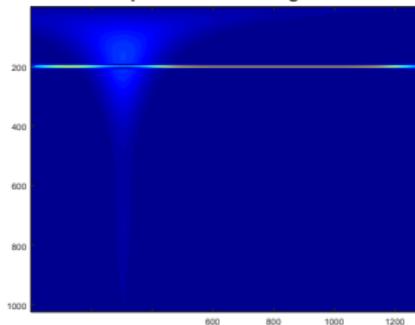


## On° continues – ex. 9 : un sinus avec perturbation locale

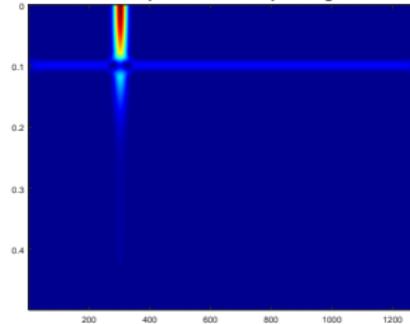
sinus avec perturbation- scalogram de Morlet



sinus avec perturbation- scalogram de Morlet



sinus avec perturbation- spectrogramme

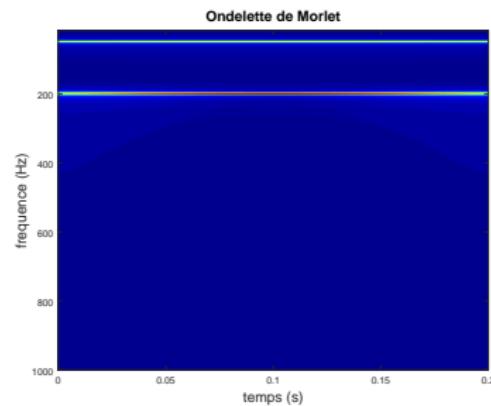
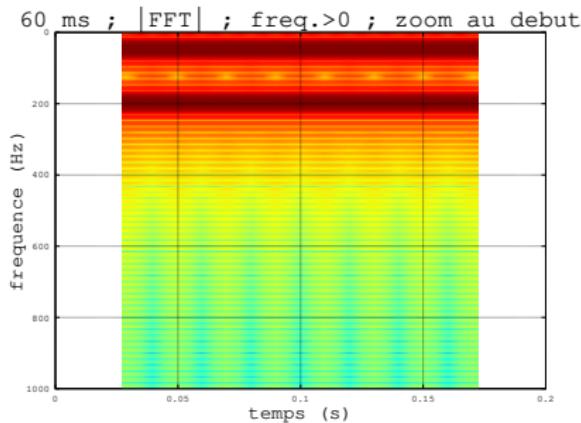


## Transformée en ondelettes continue

- ▶ Pas complètement satisfait d'avoir à modifier un outil existant, j'ai reprogrammé mon propre code, en plus simple
- ▶ Dans mon code (voir slides suivants), je considère les fréquences ainsi :
  - ▶ la fréquence à l'échelle  $a$  est la fréquence pour laquelle la réponse en fréquence du filtre passe-bande correspondant à l'échelle  $a$  passe par son maximum
- ▶ Et, de plus, moi j'étudie des signaux dans le domaine audible
- ▶ Ci-dessous, j'étudie les signaux audios de la première partie : je compare le spectrogramme et le scalogramme

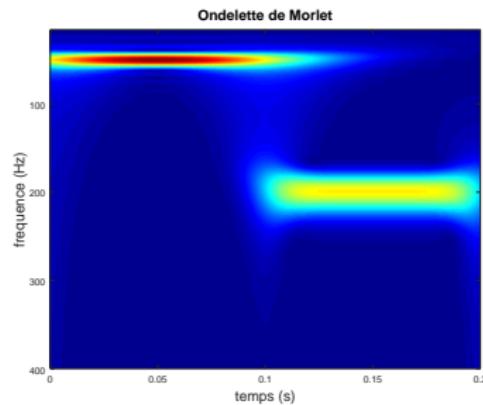
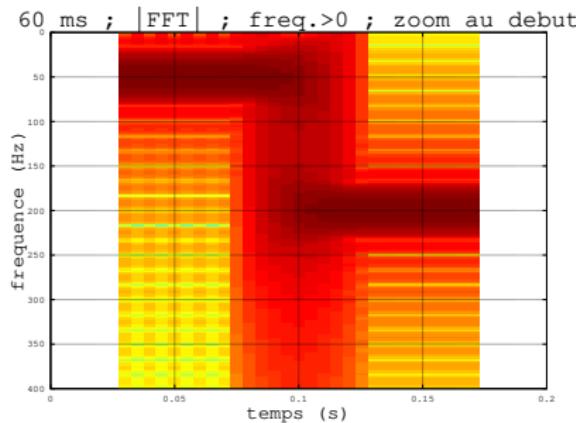
## Transformée en ondelettes continue – Mes programmes

Voir ondelettes5.m ; exemple 1 (2 sinus ensemble) :



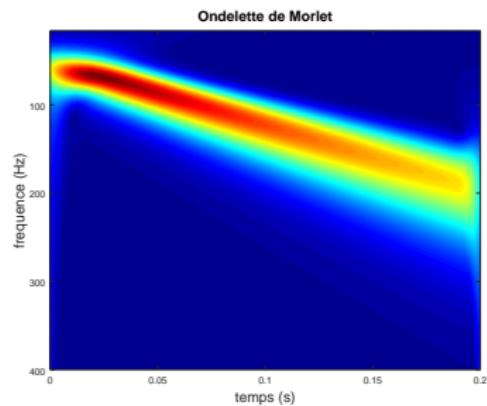
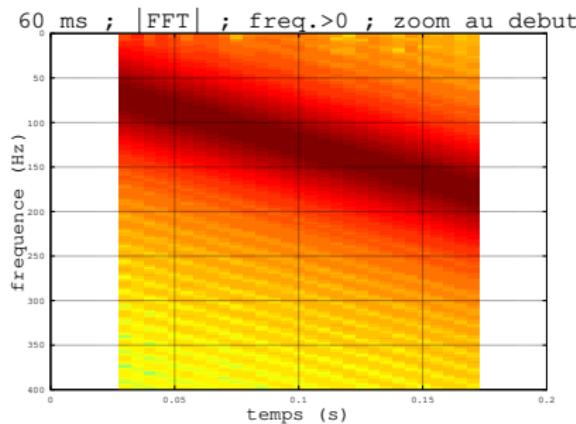
## Transformée en ondelettes continue – Mes programmes

Voir ondelettes5.m ; exemple 2 (2 sinus consécutifs) :



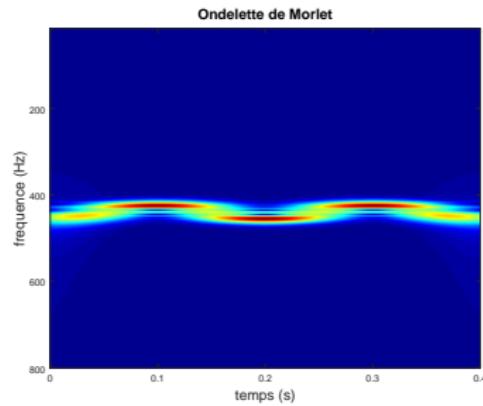
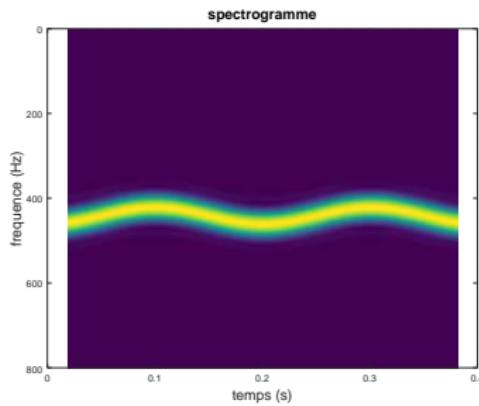
## Transformée en ondelettes continue – Mes programmes

Voir ondelettes5.m ; exemple 3 (chirp) :



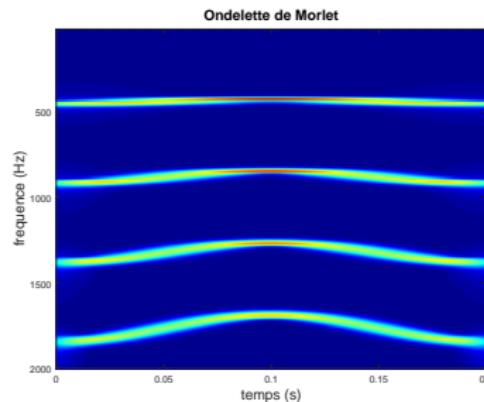
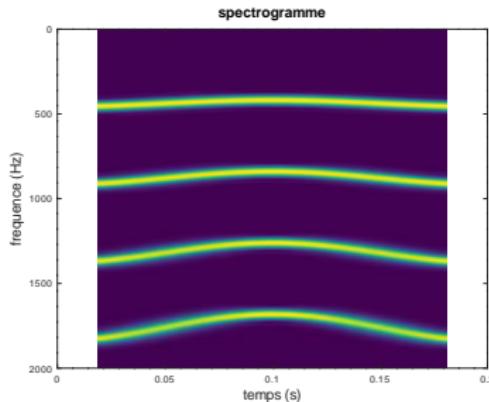
## Transformée en ondelettes continue – Mes programmes

Voir ondelettes5.m ; exemple 6 (un sinus avec vibrato) :



## Transformée en ondelettes continue – Mes programmes

Voir `ondelettes5.m`; exemple 7 (quatre sinus avec vibrato : ça ressemble beaucoup à de la voix chantée, à l'écoute) :



- ▶ je zoomé BEAUCOUP sur les fréquences les plus petites
- ▶ décimer pour se mettre à égalité avec l'exemple "tftb" pas possible : pour la voix chantée, il y a 10, 20... partiels harmoniques présents

## Transformée en ondelettes continue – Mes programmes

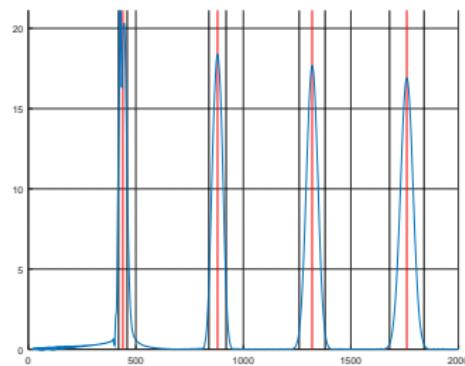
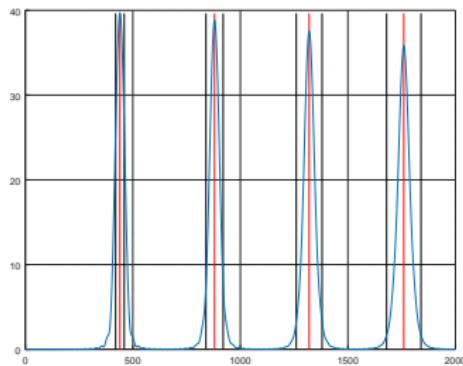
- ▶ Accord entre la fréquence théorique pour le scalogramme et la fréquence que je trouve :
  - ▶ j'inspecte la différence absolue max entre la fréquence théorique et la fréquence que je trouve
  - ▶ pour les exemples ci-dessus, le max obtenu est : 0,17 pour cent
- ▶ Pour la scalogramme, il faut régler la taille de l'ondelette (le paramètre "taille" de mes programmes, ou "wave" de "tftb")
  - ▶ comment faire ?
  - ▶ de plus, la taille de la fenêtre d'analyse dépend de l'échelle (de la fréquence)
  - ▶ pour le spectrogramme, c'est la taille des trames (environ 40 ms pour les sons) qu'il faut régler : mais on a un critère (presque) objectif : la stationnarité

## Transformée en ondelettes continue – Mes programmes

- ▶ Selon la colormap utilisée, on peut avoir des impressions visuelles différentes : mais ce ne sont que des impressions
  - ▶ note : l'amplitude est représentée, peu finement, par la couleur
- ▶ Ça peut être plus sage d'inspecter ce qui se passe spectre par spectre
  - ▶ note : par contre, on n'a pas une vue globale, sur tout le son/signal, de ce qui se passe

## Transformée en ondelettes continue – Mes programmes

Pour le dernier signal, voilà un exemple de spectre “FFT” et de spectre “ondelettes” :



⇒ c'est mieux de les voir s'animer : voir ondelettes5.m



## Transformée en ondelettes continue

- ▶ Une complication vient de la transformée inverse : elle n'existe peut-être pas (en tout cas de façon facilement et rapidement calculable)
- ▶ Note 1 : la transformée de Fourier a bien sûr une transformation inverse  $\Rightarrow$  c'est-à-dire que l'on peut reconstruire **parfaitement** le signal de départ après transformation (et elle est rapide !)
- ▶ Note 2 : par exemple, les méthodes d'analyse spectrale dites haute-résolution n'ont pas forcément une transformée inverse **parfaite** (modèles auto-régressifs, MUSIC/ESPRIT...) et/ou elle n'est pas facilement et rapidement calculable

## Transformée en ondelettes continue

- ▶ Quand on analyse un signal, on n'a pas besoin obligatoirement de le reconstruire après coup (note : moi, c'est mon cas, en général)
- ▶ Mais pour certaines applications, il faut pouvoir reconstruire le signal après coup (c'est-à-dire après transformation et traitement) :
  - ▶ pour le débruitage des signaux : sons, images...
  - ▶ parfois, pour le codage (c'est-à-dire, réduction de la taille du signal en entrée) : sons, images
  - ▶ etc.
- ▶ Et, du coup, il a fallu développer ce qu'on a appelé la transformée en ondelettes discrète

## Ondelettes discrètes

Et, avec la transformation en ondelettes discrète :

- ▶ par contre, on va avoir du mal à garder en place la notion de « fréquence » : on va la remplacer par celle d'« échelle »
- ▶ l'analyse (du contenu fréquentiel par exemple) va être plus difficile
  - ▶ pour les sons harmoniques par exemple, on peut vouloir trouver  $f_0$  (appelé aussi le pitch, la fréquence fondamentale)
  - ▶ et pour ces mêmes sons, on voudrait suivre les harmoniques (appelés aussi les partiels)

## Ondelettes discrètes – Théorie (simplifiée)

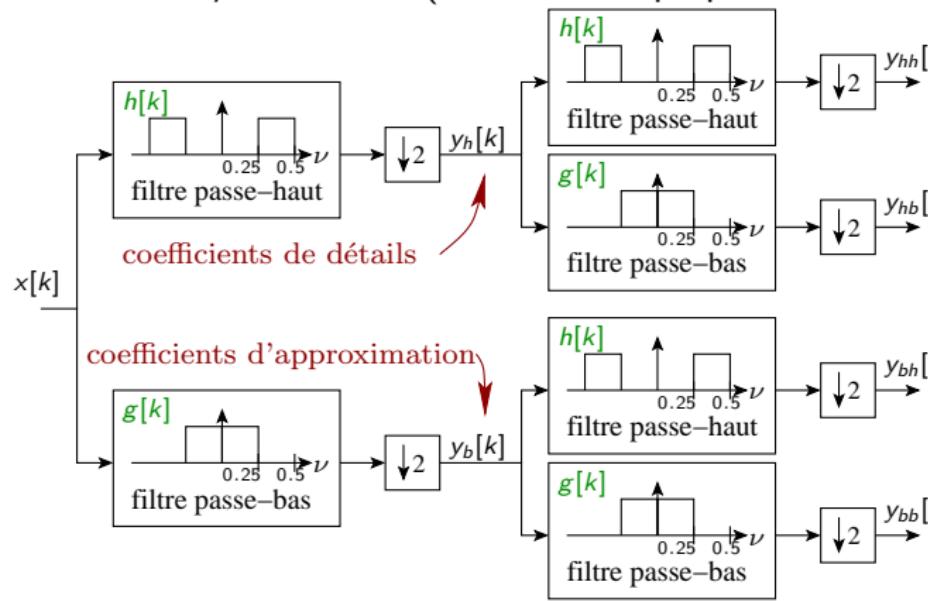
*Note : c'est un cours d'application, alors je vais limiter la théorie au maximum (je pourrais y passer tout mon temps)*

- ▶ Il s'agit de faire passer le signal par une série de filtres passe-haut ( $h[k]$ ) et de filtres passe-bas ( $g[k]$ )
- ▶ C'est lié aux bancs de filtres qu'on a déjà vus
- ▶ Par contre, les filtres utilisés sont un peu particuliers, comme on va le voir (et ils assurent qu'une transformation inverse et c'est pas rapide à calculer existe)

du tout Les réponses impulsionnelles  $h[k]$  et  $g[k]$  des filtres sont comme projeté sur Fourier, ça ressemble plus à Hilbert dérivées d'une ondelette mère  $\Psi$ , avec un facteur d'échelle  $a$  comme pour les ondelettes continues, mais cette fois-ci  $a$  n'est pas quelconque mais est une puissance de 2 (et, aussi, on décime les signaux de sortie).

# Ondelettes discrètes – Théorie (simplifiée)

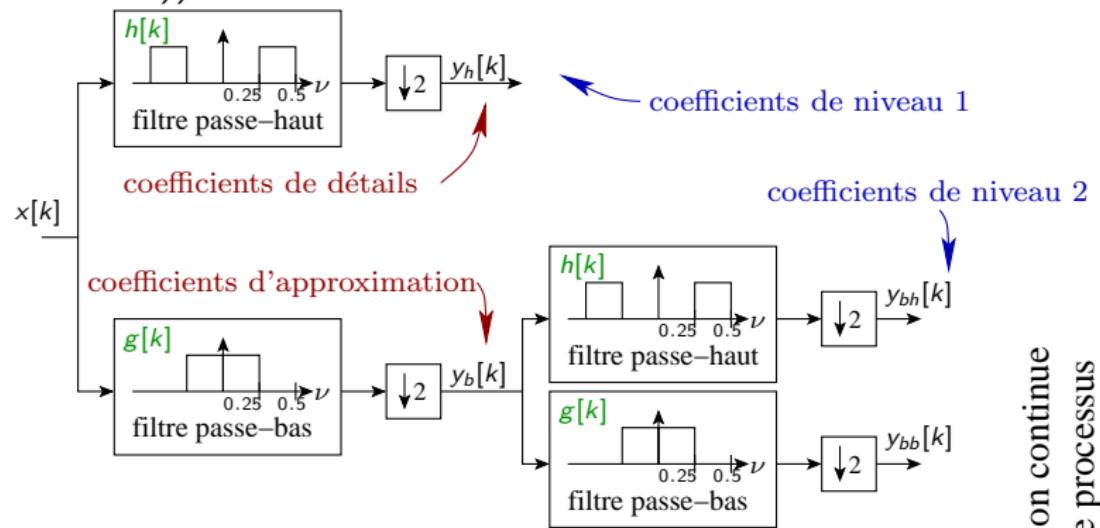
Première façon de faire (ce sont les “paquets d’ondelettes” ) :



et on continue le processus

## Ondelettes discrètes – Théorie (simplifiée)

Seconde façon de faire (c'est la "transformée en ondelettes" (plus courante)) :



## Ondelettes discrètes – Théorie (simplifiée)

Les filtres passe-haut se dérivent directement de l'ondelette mère ( $j$  est le niveau de décomposition) :

$$h^j[k] = \frac{1}{\sqrt{2^j}} \Psi\left(\frac{k}{2^j}\right)$$

Les filtres passe-bas sont plus difficiles à obtenir. Il faut pas mal batailler avec des mathématiques un peu tordues pour arriver finalement à dire que :

$$g^j[k] = -1^{-1-k} h^j[1 - k]$$

Et ça assure que nos filtres passe-bas et passe-haut auront les bonnes propriétés, notamment pour la reconstruction. (Note : nous, en tant que traiteur des signaux, ce sont les expressions des réponses impulsionnelles des filtres qui nous intéressent.)

## Ondelettes discrètes – Théorie (simplifiée)

Par contre, toute fonction ne peut pas être une ondelette mère :

- ▶ Il faut que les filtres aient un support compact, c'est-à-dire peu de coefficients non nuls (résolution)
- ▶ Il faut que les ondelettes obtenues soient orthogonales (comme pour Fourier)
- ▶ Il faut de bonnes propriétés d'approximation ; ça veut dire en particulier, des moments nuls : et alors, pour un signal régulier par morceau, on a la meilleure parcimonie (soit, le plus petit nombre possible de coefficients non négligeables)
  - ▶ les moments ( $E[X^p]$ ) sont des indicateurs de la dispersion de  $X$
- ▶ Il faut que les calculs soient rapides

## Ondelettes discrètes – Exemple : ondelette de Haar

- ▶ L'ondelette mère :
  - ▶  $\Psi(t) = 1 \quad \forall t \in 0 \leq t < 0.5$
  - ▶  $\Psi(t) = -1 \quad \forall t \in 0.5 \leq t < 1$
  - ▶  $\Psi(t) = 0$  sinon
- ▶ De ce fait pour le filtre passe-haut :
  - ▶  $h[-1] = 1/\sqrt{2}\Psi(-1/2) = 0$
  - ▶  $h[0] = 1/\sqrt{2}\Psi(0) = 1/\sqrt{2}$
  - ▶  $h[1] = 1/\sqrt{2}\Psi(1/2) = -1/\sqrt{2}$
  - ▶  $h[2] = 1/\sqrt{2}\Psi(1) = 0$ , etc.
- ▶ Et pour le filtre passe-bas :
  - ▶  $g[-1] = (-1)^0 h[2] = 0$
  - ▶  $g[0] = (-1)^{-1} h[1] = -h[1] = 1/\sqrt{2}$
  - ▶  $g[1] = (-1)^{-2} h[0] = h[0] = 1/\sqrt{2}$
  - ▶  $g[2] = (-1)^{-3} h[-1] = 0$ , etc.

## Ondelettes discrètes – Exemple : Daubechies

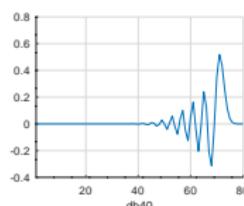
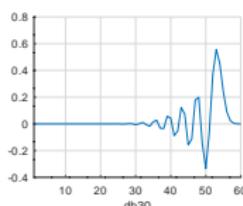
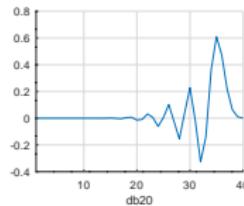
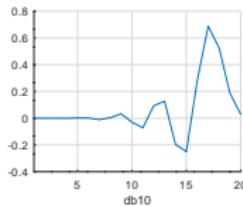
ondelette de Haar très utilisée

- ▶ La famille d'ondelettes de Daubechies est vaste
- ▶ Elle part du théorème (de Daubechies) suivant : si une ondelette a  $p$  moments nuls, alors  $h$  et  $g$  ont au moins  $2p$  coefficients non nuls (et  $\Psi$ , l'ondelette, a un support de taille au moins  $2p - 1$ )
- ▶ Les ondelettes de Daubechies atteignent cette borne
  - ▶ les ondelettes de Daubechies, parce qu'il y a une ondelette par ordre  $p$  : ainsi, dans les outils disponibles, on va de 1 à plusieurs dizaines
  - ▶ Note : et les autres familles d'ondelettes trouvent un compromis entre nombre de moments nuls et taille du support

## Ondelettes discrètes – Exemple : Daubechies

Filtres  $h$  de Daubechies pour divers ordres :

spécialement  
concues pour  
atteindre p  
moments nuls



⇒ la plage d'ordres disponibles est grande ; et ceci vaut pour toutes les familles d'ondelettes

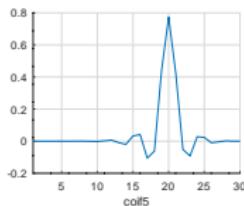
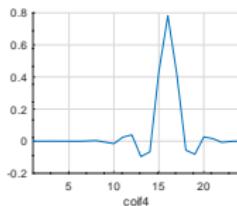
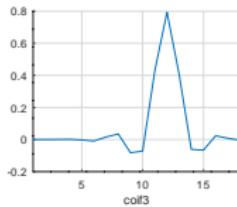
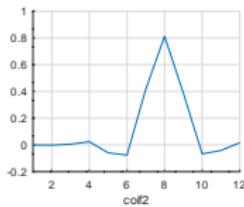
## Ondelettes discrètes – Exemples

Il y a tout un tas de familles d'ondelettes :

- ▶ L'outil “Itfat” nous donne accès à toutes ces ondelettes :  
algmband, cmband, coif (Coiflets), db (Daubechies), dden,  
ddena, ddenb, dgrid, hden, lemarie, mband, oddevena,  
oddevenb, optsyma, optsymb, qshifta, qshiftb, remez, spline,  
sym, symdden, symds, symorth, symtight
- ▶ La *wavelet toolbox* de matlab propose ces familles :  
Daubechies, Coiflets, Symlets, Fejér-Korovkin, Discrete Meyer,  
Biorthogonal, Reverse Biorthogonal

## Ondelettes discrètes – Exemple : Coiflets/Coifman

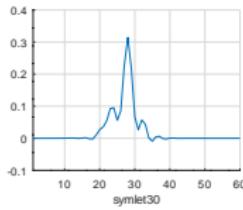
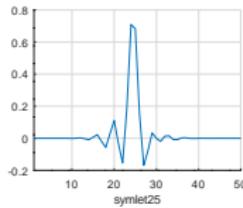
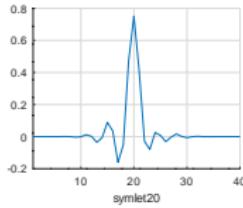
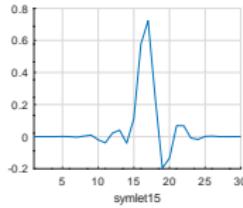
Filtres  $h$  de Coifman pour divers ordres :



⇒ particularités : plus proche de la symétrie que Daubechies, mais borne de Daubechies pas atteinte

## Ondelettes discrètes – Exemple : Symlets

Filtres  $h$  “symlets” pour divers ordres :



⇒ particularités : la symétrie est “presque” atteinte (mais pas tout à fait)

## Ondelettes – Outils

- ▶ Pour ceux qui utilisent matlab, il y a des fonctions dédiées ;

- ▶ voir la *wavelet toolbox* :

<https://fr.mathworks.com/products/wavelet.html>

- ▶ ou *WaveLab* : ([statweb.stanford.edu/~wavelab/](http://statweb.stanford.edu/~wavelab/))

<https://github.com/gregfreeman/wavelab850?tab=readme-ov-file>

- ▶ Pour ceux qui utilisent octave (et/ou matlab, sans avoir la *wavelet toolbox*) voir :

- ▶ “ltfat” (the Large Time/Frequency Analysis Toolbox) :

[ltfat.github.io](https://ltfat.github.io); la doc, **ltfat.pdf**, est utile

- ▶ “tftb” (Time-Frequency Toolbox) : doc **refguide.pdf** utile ;  
[http://tftb.nongnu.org/index\\_fr.html](http://tftb.nongnu.org/index_fr.html)

- ▶ Pour ceux qui utilisent python, voir Scipy.signal, PyWavelets, autres ?

## Ondelettes bi-orthogonales

- ▶ On a vu que les ondelettes de Daubechies ne sont pas symétriques
- ▶ C'est dû au deux contraintes dont j'ai parlé : support compact et orthogonalité
- ▶ Si on lève la contrainte d'orthogonalité, on arrive à construire des ondelettes symétriques
- ▶ La symétrie est importante pour réduire le nombre d'artefacts introduits par la transformation ("artefacts" : du type lobes secondaires pour Fourier, etc.)

## Ondelettes bi-orthogonales

- ▶ Entre les ondelettes orthogonales et les ondelettes bi-orthogonales, l'autre différence importante intervient en ce qui concerne la reconstruction du signal (la transformation inverse)
- ▶ Dans le cas des ondelettes orthogonales les filtres sont identiques pour la transformation et la transformation inverse
- ▶ Ce n'est pas le cas pour les ondelettes bi-orthogonales
- ▶ Notez que ça va être transparent pour vous dans la suite

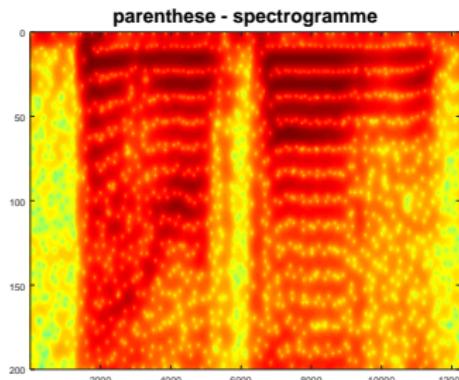
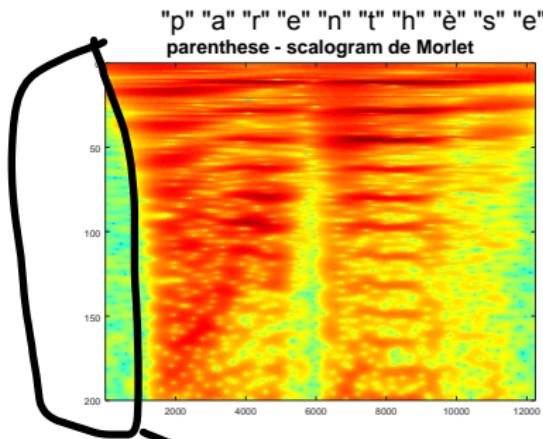
## Bancs de filtres versus Ondelettes

- ▶ Les ondelettes sont des bancs de filtres à reconstruction parfaite
- ▶ Par contre, tous les bancs de filtres à reconstruction parfaite ne sont pas des ondelettes
  - ▶ il peut y avoir des problèmes d'instabilité quand le nombre de niveaux de décomposition augmente
  - ▶ note : a priori, c'est transparent pour vous, mais il faut savoir que ça se passe comme ça

## Application 1 : analyse son ("parenthese.wav")

Voir "scaloversusperio.m" ; mon scalogram versus le spectrogramme :

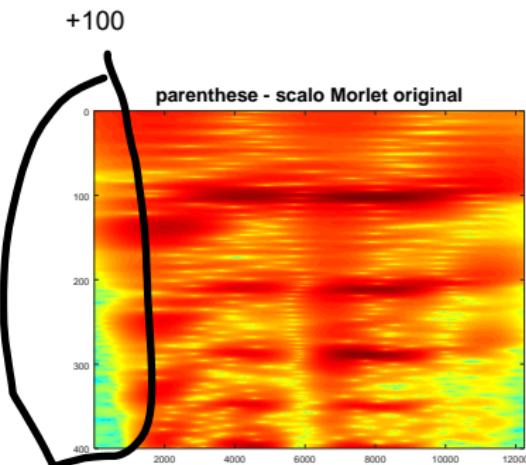
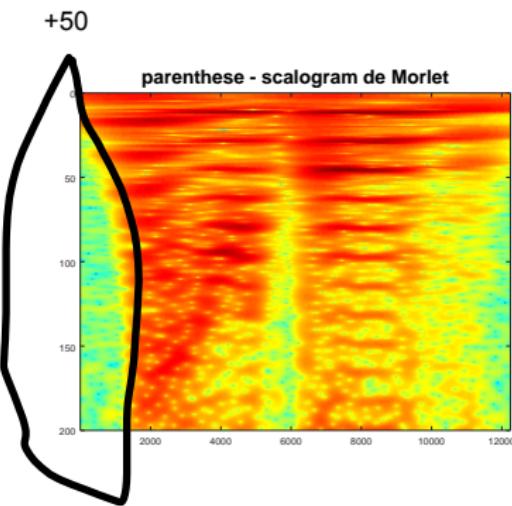
scalogram de morlet permet d'être plus précis



numéro de coefficient de l'ondelette

## Application 1 : analyse son (“parenthese.wav”)

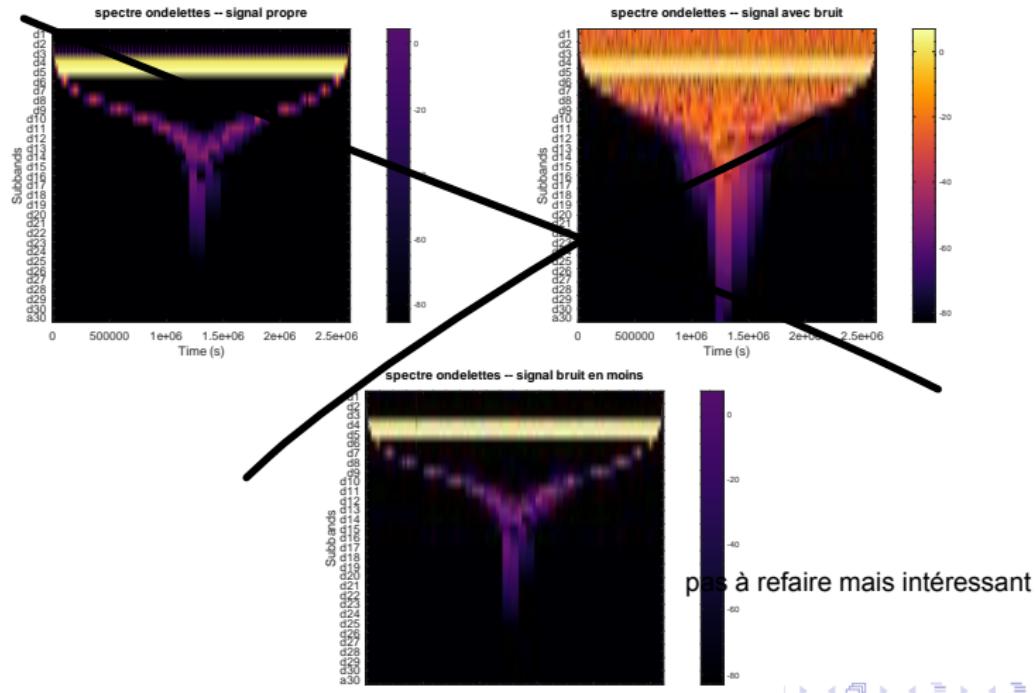
Voir “scaloversusperio.m” ; mon scalogram versus le scalogram original :



## Application 2 : débruitage d'un signal avec les ondelettes

- ▶ On considère l'exemple simple d'un sinus à 500 Hz noyé dans du bruit (de plus, on est dans le domaine audible)
- ▶ On fait la décomposition en ondelettes
- ▶ Le bruit est blanc, donc son énergie doit se répartir sur tous les coefficients en ondelettes, à tous les niveaux (comme ça se passe avec Fourier)
- ▶ En mettant à 0 tous les plus petits coefficients, on espère éliminer le bruit
- ▶ Voir (et écouter) ce qui se passe avec **debruitage1.m** ; on entend d'abord le signal bruité, puis le signal après débruitage

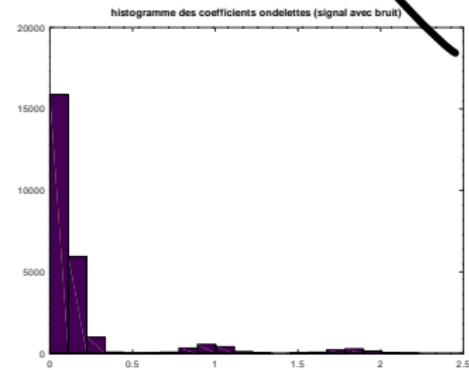
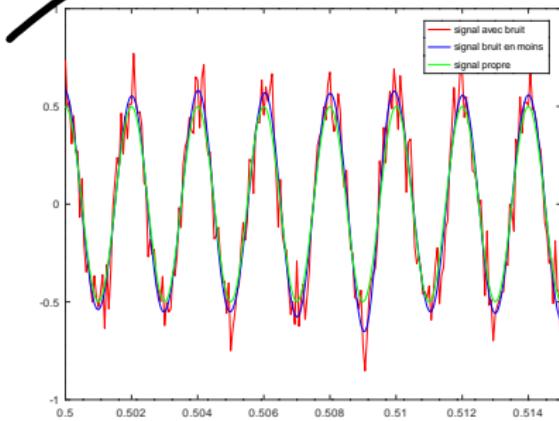
## Application 2 : débruitage ; “scalogrammes” obtenus



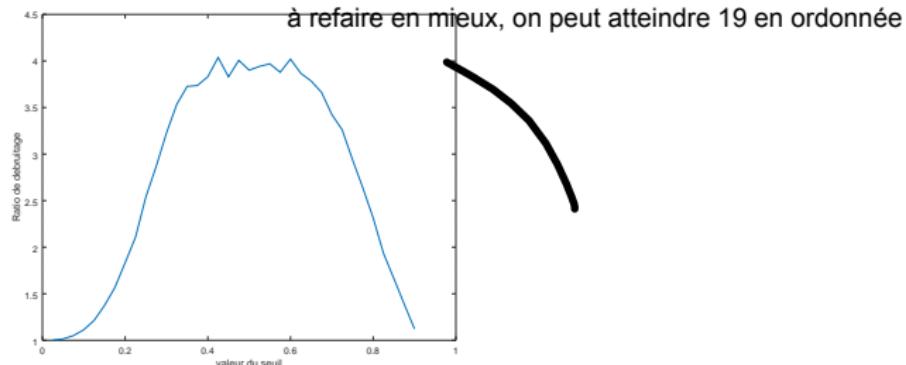
## Application 2 : débruitage ; signaux obtenus

Signaux à gauche ; et à droite histogramme des modules des coefficients d'ondelette

ceux là sont à refaire



## Application 2 : débruitage ; optimisation du seuil



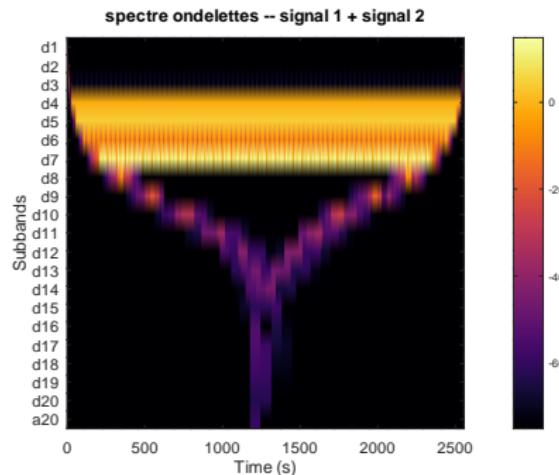
- ▶ Voir **débruitage2.m**
- ▶ Mesure du rapport des énergies, de la différence entre le signal bruité et le signal propre, et de la différence entre le signal débruité et le signal propre
- ▶ Notez : statistiques sur un très grand nombre d'essais, etc.
- ▶ On attend de vous cette attitude pour les TLs

## Application 3 : séparation de sources avec les ondelettes

- ▶ On considère l'exemple simple d'une somme de deux sinus à 100 Hz et 500 Hz (de plus, on est dans le domaine audible)
- ▶ On fait la décomposition en ondelettes
- ▶ Normalement, chacun des sinus n'occupe qu'une partie des niveaux de décomposition (comme ça se passe avec Fourier)
- ▶ En coupant le spectre en deux parties, et en mettant à 0 tous les coefficients d'ondelette qui manquent, on espère séparer les deux sinus
- ▶ Voir (et écouter) ce qui se passe avec **separation1.m** ; on entend d'abord le signal initial, puis chacun des sinus obtenus

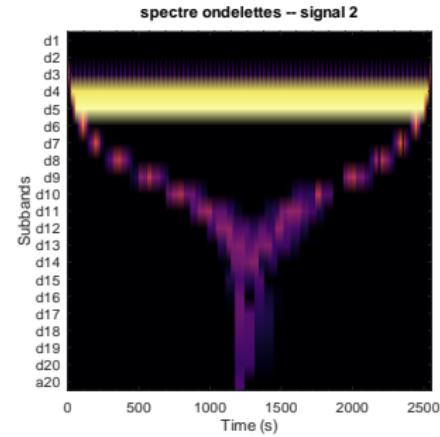
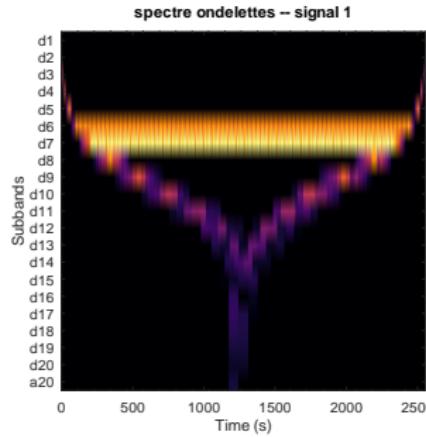
## Application 3 : séparation de sources avec les ondelettes

“Scalogramme” de la somme des deux sinus :



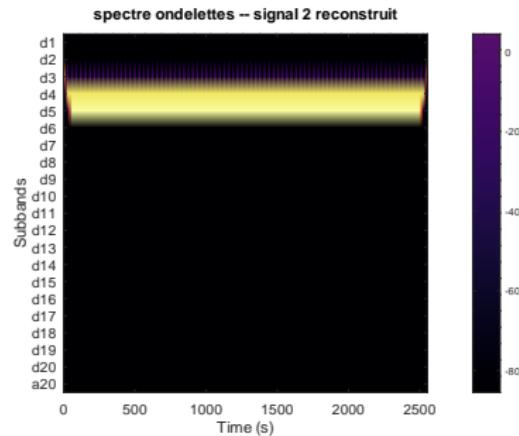
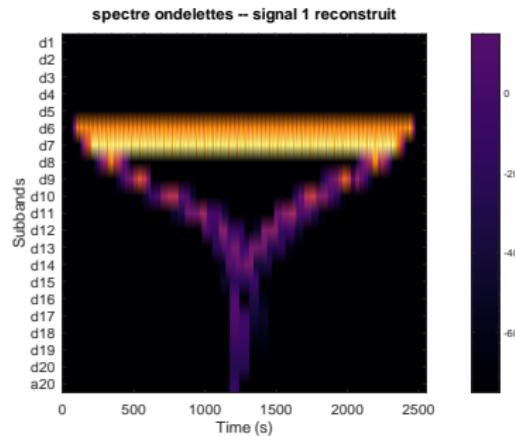
## Application 3 : séparation de sources avec les ondelettes

“Scalogrammes” de chacun des sinus, séparément (avant leur somme) :



## Application 3 : séparation de sources avec les ondelettes

“Scalogrammes” de chacun des sinus, après séparation :



## Application 3 : séparation de sources avec les ondelettes

- ▶ Ça semble être efficace (à l'audition) ; voir aussi les deux erreurs données par le code ; note : certains coefficients, mis à 0 alors qu'ils apparaissent en couleur, sont suffisamment petits pour que ça n'ait pas d'impact
- ▶ Une difficulté, ça peut être de déterminer quels coefficients d'ondelette correspondent à quel niveau de décomposition (voir le code)
- ▶ Il faut remarquer que les ondelettes, telles qu'utilisées ici, ne sont pas forcément adaptées à la séparation de sources dans le domaine audio
- ▶ Il faudrait sans doute mieux utiliser les paquets d'ondelettes par exemple

## Sujet de TL – Partie 1 (suite)

- ▶ 1. Reproduisez la décomposition en banc de filtres de la partie précédente
  - ▶ Ceci dans le langage que vous préférez : Matlab/Octave/Python
  - ▶ aide en ligne de matlab (tapez “filtering”, par exemple) :  
<https://fr.mathworks.com/help/matlab/index.html>
  - ▶ pour python : numpy, matplotlib, scipy.signal semblent utiles ; et regardez ce site, notamment, où il y a des exemples (pour le filtrage, etc.) :

<https://www.f-legrand.fr/scidoc/docimg/sciphys/caneurosmart/pysignal/pysignal.html>

- ▶ 2. Ajoutez un vibrato d'amplitude 20 Hz et de fréquence 5 Hz au signal de départ, et commentez ce qui se passe ; augmentez l'amplitude du vibrato

## Sujet de TL – Partie 2a

Débruitage d'un signal avec les ondelettes :

- ▶ Essayez de reproduire l'exemple donné en cours
- ▶ Essayez avec plusieurs ondelettes
- ▶ Jouez sur le nombre de décompositions
- ▶ Jouez sur les autres paramètres disponibles (seuil, variance du bruit, etc.)
- ▶ Essayez de régler automatiquement le seuil : seuil universel de Donoho, etc. (il y a des dizaines de seuils automatiques)

## Sujet de TL – Partie 2b

Séparation de sources avec les ondelettes :

- ▶ Essayez de reproduire l'exemple donné en cours
- ▶ Essayez avec plusieurs ondelettes
- ▶ Jouez sur le nombre de décompositions
- ▶ Jouez sur les autres paramètres disponibles (seuil, fréquences des deux sinus...)
- ▶ Essayez de déterminer automatiquement le seuil

## Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ **Partie 4 : Notions d'apprentissage supervisé**
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ Partie 7 : Conclusion

## L'apprentissage supervisé recouvre quoi ?

⇒ Définition trouvée sur le net :

*Raisonnement mathématique qui permet d'analyser et d'interpréter un gros volume de données, de différentes sources, afin de dégager des tendances, de rassembler les éléments similaires en catégories statistiques et de formuler des hypothèses.*

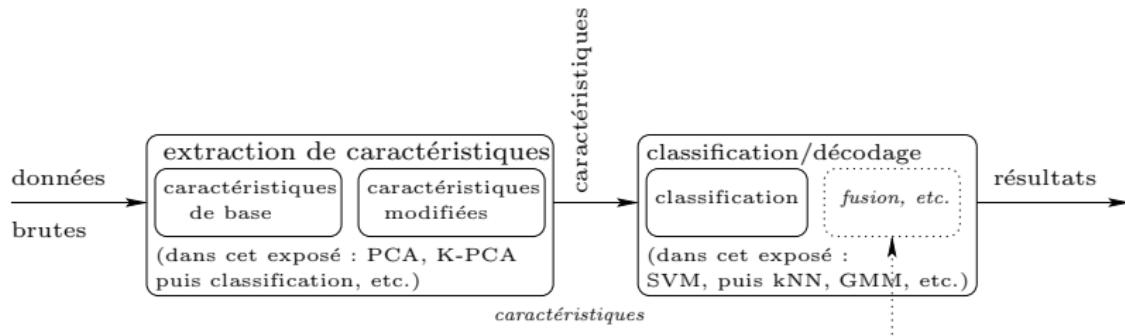
⇒ Exemples :

- ▶ Classification
- ▶ Régression, linéaire ou non
- ▶ Détection de données anormales
- ▶ Détermination de composantes principales
- ▶ Etc.

⇒ Disons, pour nous ici, que c'est de la classification

# L'apprentissage supervisé – Notions

## Chaîne de traitement classique :



⇒ Les méthodes à noyaux interviennent/peuvent intervenir dans les deux étapes ; elles vont me servir de fil conducteur

## L'apprentissage supervisé – Notions

**Et l'homme apporte la connaissance qu'il a du problème, ce à 3 niveaux :**

1. Au niveau du choix des caractéristiques à extraire.
  - ▶ Au moins : comment présenter les données au classifieur (échantillonnage, découpage en trames, etc.).
  - ▶ Petit exemple (très simpliste) : en reconnaissance des images, si l'on veut différencier la classe des *chats* de celle des *chiens*, il ne semble pas très intéressant de prendre comme caractéristique le nombre de pattes ; par contre, cette caractéristique peut être utile pour différencier les *chats* des *oiseaux*.
  - ▶ Note : le deep-learning résoud pour certains problèmes le deuxième point, mais pas le premier.

## L'apprentissage supervisé – Notions

2. Au niveau de la classification, notamment en entraînant le classifieur à partir d'entités qu'il a auparavant annotées à la main. Puis les données sont séparées en 2 groupes. Elles sont rangées dans un groupe ou dans l'autre de façon aléatoire.
  - Le premier (groupe « apprentissage »), comprenant  $m$  entités, sert à entraîner le classifieur : la fonction de décision est déterminée à partir de ce groupe.
  - Les entités du second groupe (groupe « test ») sont classées en utilisant cette fonction de décision. Une première validation du classifieur est faite.
  - Idéalement, un 3ème groupe (groupe « validation croisée ») est utilisé pour finir de valider le classifieur. Ce groupe est constitué de données ne provenant pas des groupes de données initiaux, mais obtenues par exemple dans des conditions différentes.

## L'apprentissage supervisé – Notions

3. Au niveau de la détection, notamment dans les problèmes séquentiels, en indiquant quelles séquences sont possibles, et quelles séquences sont impossibles ; et quelles séquences sont plus probables que d'autres. Par exemple, en reconnaissance de la parole, en français, certaines séquences de phonèmes ne sont pas possibles ; et la séquence de phonèmes « [l][e] » est plus fréquente que la séquence de phonèmes « [l][i] ».

⇒ Après toute cette procédure, on peut utiliser le système en aveugle, comme une boîte noire.

## L'apprentissage supervisé – Exemple

**Données brutes :** Une personne écrit des lettres sur une tablette graphique et on veut reconnaître ce qu'elle écrit. Les données brutes qu'on mesure sont : 1. à une certaine fréquence d'échantillonnage, la position de la pointe du crayon sur la tablette ; 2. et les instants où la pointe quitte ou rejoint la tablette. ⇒ pour simplifier, les instants 2. sont obtenus avec un petit retard temporel, pour que ces instants correspondent à des instants d'échantillonnage, ce qui permet que le signal soit échantillonné régulièrement : ceci rend les traitements ultérieurs plus aisés.

## L'apprentissage supervisé – Exemple

⇒ Pour simplifier ENCORE, considérons de plus qu'on a donné des instructions à cette personne, telles que :

- ▶ les lettres sont séparées les unes des autres par un petit espace blanc : elle doit lever son crayon entre deux lettres.
- ▶ le scripteur ne lève jamais son crayon au cours de l'écriture d'une lettre donnée. Ce qui implique par exemple qu'il ne DOIT pas mettre les points sur les « i » et les « j »; et qu'il ne forme pas les barres des « t ».
- ▶ les lettres sont écrites en minuscule

On a donc un problème de reconnaissance de 26 classes, c'est-à-dire de classification de chacune des entités formées par le scripteur dans une des 26 classes possibles.

## L'apprentissage supervisé – Exemple

**Caractéristiques :** Il faut tout d'abord extraire des caractéristiques des données brutes, caractéristiques qui vont permettre :

- ▶ Premièrement de réduire la dimensionnalité des données mesurées. Le signal d'écriture est typiquement échantillonné à 250 Hz, c'est-à-dire que chaque lettre est composée de plusieurs dizaines d'échantillons, chaque échantillon ayant lui-même deux dimensions : communément, la position en  $x$  et la position en  $y$  (d'autres dimensions, comme la pression et l'inclinaison du crayon dans deux directions, sont très souvent prises en compte). Appelons  $D$  la dimension des données brutes.

## L'apprentissage supervisé – Exemple

- ▶ Et deuxièmement de mettre en relief  $N$  (avec  $N < D$ ) traits distinctifs de chacune des classes. Par exemple, un « o » a plutôt (ce n'est pas exact pour tous les scripteurs) une forme arrondie, alors qu'un « l » a plutôt une forme allongée. Aussi, pour chaque entité formée, une caractéristique à considérer pourrait être le rapport de son extension suivant l'axe des  $x$  et de son extension suivant l'axe des  $y$ .

## L'apprentissage supervisé – Exemple

### **Classification :**

Aussi, comme indiqué ci-dessus,  $N$  caractéristiques sont extraites à partir des données brutes. Ces données d'observation, quand on les range dans un espace à  $n$  dimensions et disons euclidien, s'organisent en 26 nuages de points correspondant chacun à une classe.

⇒ Pour simplifier, on considère que ces nuages ne se mélangent autant que possible pas.

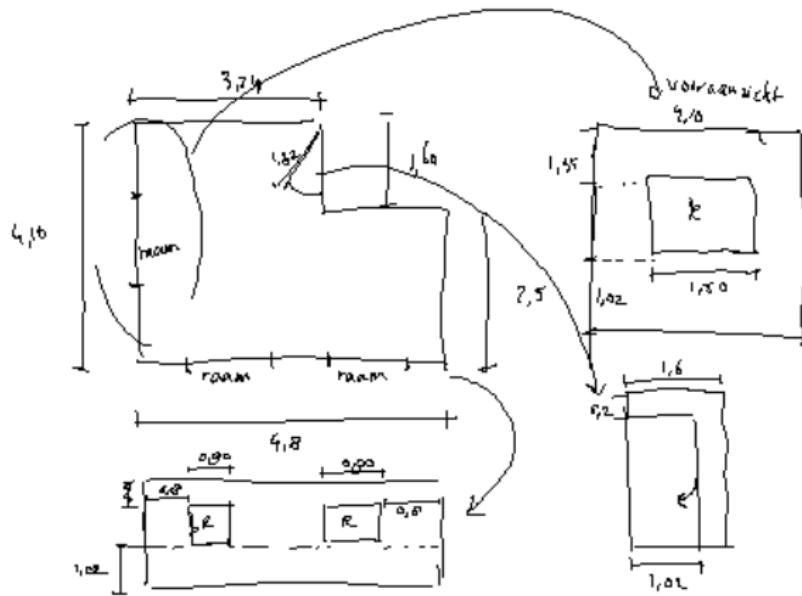
## L'apprentissage supervisé – Exemple

**Détection :** Mais les nuages se mélangent un peu. À la base, les « o » et les « I » sont composés d'une boucle, d'où une confusion aisée entre les deux.

Le classifieur donne, pour une certaine entité à classer, comme hypothèse la plus probable que c'est un « I », comme deuxième hypothèse que c'est un « o », et comme troisième, quatrième, etc. hypothèse... Mais on sait, avec certitude ( $\Rightarrow$  pour simplifier), que l'entité précédente est un « q ». En français, la séquence « ql » n'est pas possible, alors il faut corriger la sortie de la classification, c'est-à-dire ne pas choisir la première hypothèse mais la suivante.

L'étape de détection est là pour résoudre ce type de problèmes.  
...et dès lors tout devrait bien se passer...

## L'apprentissage supervisé – La (dure) réalité



## Notions de base sur l'apprentissage supervisé

Caractéristiques/Traitements (vers la parcimonie)

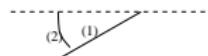
Classification/Apprentissage/Test/Validation croisée

Rappel pour le TL : risque empirique et risque espéré

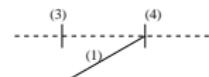
# L'apprentissage supervisé – La (dure) réalité



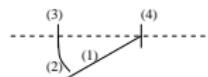
The simplest door: one stream (D)



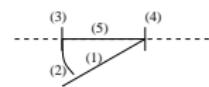
Two streams (D)



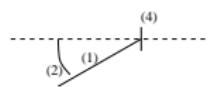
More complicated door:  
three streams (Mak)



More complicated door:  
four streams (Mak)



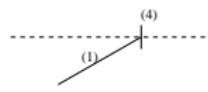
More complicated door:  
five streams (Mak)



More complicated door:  
three streams (Mir)



Different door:  
two streams + word "deur" (Mir)



Different door:  
two streams without word (Mir)

## PCA et K-PCA, présentation

But(s) de la PCA (Principal Component Analysis) et de la K-PCA :

- ▶ réduction de la dimensionnalité :
  - + ne garder que les caractéristiques les moins corrélées
  - + le nombre d'échantillons nécessaires à l'apprentissage croît exponentiellement avec leur dimensionnalité (malédiction de la dimensionnalité)
- ▶ et, pour la K-PCA seulement, améliorer les caractéristiques (dépliement d'une variété dans un espace de dimension  $L$ , variété de dimension  $l < L$ , dans un espace de dimension  $l$ )

# PCA (Principal Component Analysis)

## Ce qu'est la PCA :

- ▶ matrice d'observations  $x$   
 $\Rightarrow$  dimension  $N \times m$  :  $N$  caractéristiques et  $m$  observations par caractéristique
- ▶ pour chaque caractéristique, les observations sont centrées
- ▶ calcul de la matrice de covariance :

$$C^{(PCA)} = \frac{1}{m} \sum_{j=1}^m x_j x_j^T$$

$\Rightarrow$  dimension  $N \times N$

## PCA (Principal Component Analysis)

- ▶ calcul des  $N$  valeurs propres  $\lambda$  et des vecteurs propres  $v$  correspondants (matrice  $N \times N$ )  $\Rightarrow$  ceci revient à résoudre l'équation suivante :

$$\lambda v = C^{(PCA)} v \quad [\text{équation 1}]$$

- ▶ projection des observations sur les  $d < N$  vecteurs propres correspondants aux  $d$  plus grandes valeurs propres

$$\text{projection}_{(k)} = x^T v_{(k)} \quad k = 1 \dots d$$

- ▶ concaténation des transposées des  $d$  projections pour obtenir  $x^{(PCA)}$   $\Rightarrow$  nouvelle matrice d'observations  $x^{(PCA)}$  (de dimension  $d \times m$ )

## K-PCA – extraction de caractéristiques

### Les méthodes à noyau :

- ▶ l'idée des méthodes à noyau, c'est, via un mapping non-linéaire  $\phi$ , de faire passer les données d'observation  $x_1 \dots x_m$  dans un espace  $\mathcal{F}$  possiblement de (beaucoup) plus grande dimension, avec :

**notation 1 :**

$$\begin{array}{ccc} \phi : \mathbb{R}^N & \rightarrow & \mathcal{F} \\ x & \mapsto & \phi(x) \end{array}$$

**notation 2 :**

$$\begin{array}{ccc} k(x, \cdot) : \mathbb{R}^N & \rightarrow & \mathcal{F} \\ x & \mapsto & k(x, \cdot) \end{array}$$

- ▶ et du coup l'ensemble d'apprentissage n'est plus  $x_1, \dots, x_m$ , mais  $\phi(x_1), \dots, \phi(x_m)$  ou  $k(x_1, \cdot), \dots, k(x_m, \cdot)$

## K-PCA – extraction de caractéristiques

### La K-PCA :

- ▶ au lieu d'utiliser directement la matrice de covariance, il s'agit ici de partir de la matrice de covariance après passage dans l'« espace de transformation » :

$$C^{(K-PCA)} = \frac{1}{m} \sum_{j=1}^m \phi(x_j) \phi(x_j)^T$$

## K-PCA – extraction de caractéristiques

- (a) puis, tout d'abord, si l'on réécrit l'équation [1], du problème des valeurs propres, on a :

$$\lambda v = C^{(K-PCA)} v = \frac{1}{m} \sum_{j=1}^m \langle \phi(x_j), v \rangle \phi(x_j)$$

- (b) ensuite, il existe des  $\alpha_i$  tels que :

$$v = \sum_{i=1}^m \alpha_i \phi(x_i)$$

- (c) de plus, le « truc du noyau » dit qu'on a par définition :

$$k(x, x') = \langle \phi(x), \phi(x') \rangle$$

## K-PCA – extraction de caractéristiques

### La K-PCA :

- ▶ Du coup, en combinant (a), (b) et (c), on aboutit à l'équation équivalant à [1] pour la PCA, qui s'écrit pour la K-PCA :

$$m\lambda\alpha = \hat{K}\alpha \quad [\text{équation 2}]$$

## K-PCA – extraction de caractéristiques

- ▶ où  $\hat{K}$  est la matrice de Gram centrée (dimension  $m \times m$ ) :
  - $K$  telle que :  $K_{ij} = k(x_i, x_j)$
  - et  $\hat{K} = K - I_m K - K I_m + I_m K I_m$   
 $\Rightarrow$  ce ne sont pas les  $\phi$  qui sont centrés, puisque de toute façon ils ne sont pas calculés, mais c'est  $K$  qui doit être centrée

## K-PCA – extraction de caractéristiques

- ▶ et où  $\alpha$  (dimension  $m \times m$ ) est la matrice des vecteurs propres (équivalent à  $v$  pour la PCA) lui correspondant
- ▶ les  $d$  vecteurs propres correspondant aux  $d$  plus grandes valeurs propres forment la nouvelle matrice d'observations  $x^{(K-PCA)}$  [et pas de projection à faire]
- ▶ **Note : contrairement à ce qui se passe pour la PCA,  $d$  peut être supérieur à  $N$**

## Remarques : PCA versus K-PCA – Exemple simple

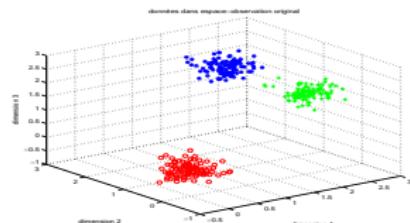


Figure – Données initiales

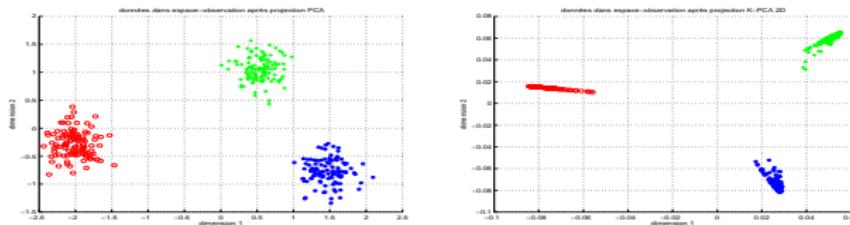


Figure – Gauche : Données après PCA. Droite : Données après K-PCA

## Remarques : Résultats concernant la K-PCA

Sur des données simulées, après avoir optimisé le code *octave*, nous obtenons, comme temps de calcul en secondes en fonction du nombre de points (c'est long : **aïe !**) :

7650 points	8976.92 s
7800 points	9583.28 s
8100 points	10579.02 s
8400 points	11770.87 s
8700 points	13135.16 s
9000 points	14371.16 s
10200 points	20998.43 s
10500 points	22845.46 s
10800 points	24811.90 s

⇒ et au-delà de 10800, “out of memory” : **aïe !**

⇒ et croissance en  $m^3$  : **aïe !**

→ Machine : processeur AMD64 2.6 GHz ; RAM : 4 Go

## Remarques : Quelques noyaux

nom	définition	paramètres
RBF gaussien	$k(x_i, x_j) = \exp\left(-\frac{\ x_i - x_j\ ^2}{2\sigma^2}\right)$	$2\sigma^2$
quadratique rationnel	$k(x_i, x_j) = \frac{\sigma - \ x_i - x_j\ ^2}{\sigma + \ x_i - x_j\ ^2}$	$\sigma$ (problème si trop petit)
polynomial	$k(x_i, x_j) = \langle x_i, x_j \rangle^p$	$p$
sigmoïdal	$k(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \vartheta)$	$\kappa, \vartheta$

## Remarques : Quelques noyaux

nom	définition	paramètres
polynomial inhomogène	$k(x_i, x_j) = (\langle x_i, x_j \rangle + c)^d$	$c, d$
exponentiel	$k(x_i, x_j) = \exp(-c  x_i - x_j  ^\beta)$	$c, 0 \leq \beta \leq 2$
multi-quadratique	$k(x_i, x_j) = -\sqrt{  x_i - x_j  ^2 + c^2}$	$c$
multi-quadratique inverse	$k(x_i, x_j) = \frac{1}{\sqrt{  x_i - x_j  ^2 + c^2}}$	$c$

## Remarques : Quelques noyaux

- ▶ Le choix du noyau, pour la résolution de tel ou tel problème, est important.
- ▶ Le choix des paramètres du noyau (comme  $\sigma$  pour le RBF gaussien) est important.
- ▶ **Attention : ces noyaux sont aussi ceux utilisés pour la classification par SVM (voir suite de l'exposé) !!!**
- ▶ Toute combinaison linéaire de noyaux est elle-même un noyau.

## Remarques : Quelques noyaux

En fait, sont des noyaux :

- ▶  $k_1(x, x') + k_2(x, x')$
- ▶  $ck_1(x, x')$  pour  $c > 0$
- ▶  $c + k_1(x, x')$  pour  $c > 0$
- ▶  $k_1(x, x')k_2(x, x')$
- ▶  $k_1(x, x')/\sqrt{k_1(x, x)k_1(x', x')}$
- ▶  $k_1(f(x), f(x'))$  où  $f$  est une fonction de  $\mathcal{X}$  dans  $\mathbb{R}$

## Topo sur la classification

- ▶ Un classifieur doit fournir une frontière avec de bonnes performances de généralisation (taux de données n'ayant pas participé à l'apprentissage de cette frontière qui se retrouvent bien classées aussi haut que possible)
- ▶ La frontière doit coller aux données d'apprentissage
- ▶ La frontière doit être lisse, ceci afin d'éviter le surapprentissage

## Topo sur la classification

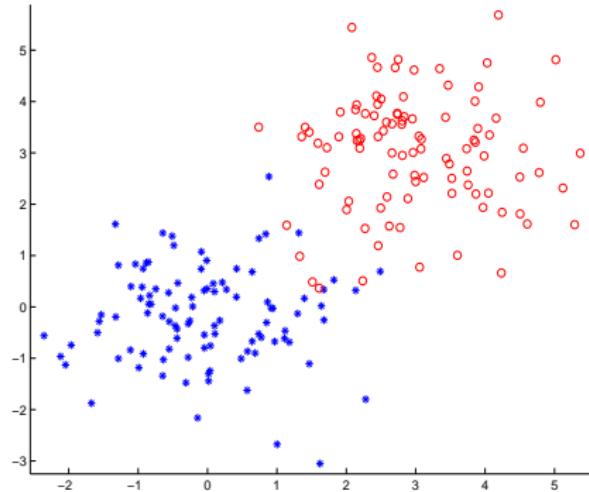


Figure – Données

## Topo sur la classification

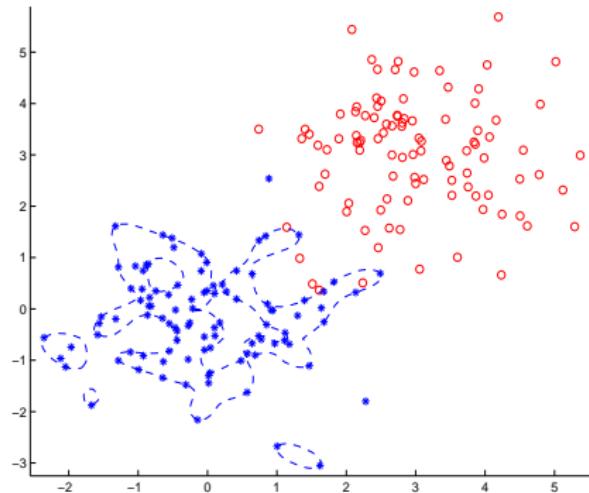


Figure – Surapprentissage, généralisation mauvaise

## Topo sur la classification

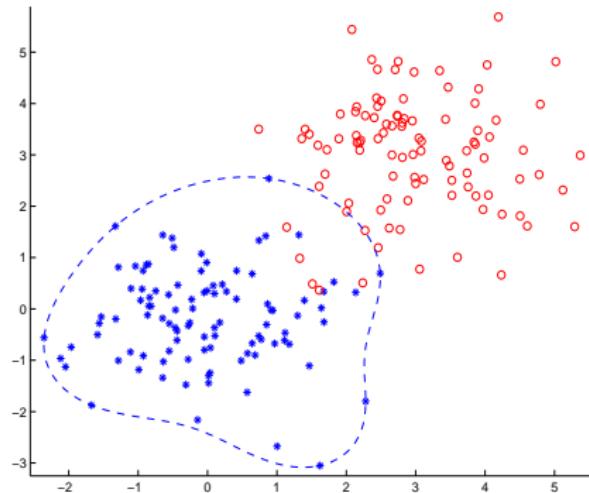


Figure – Correct

## Topo sur la classification

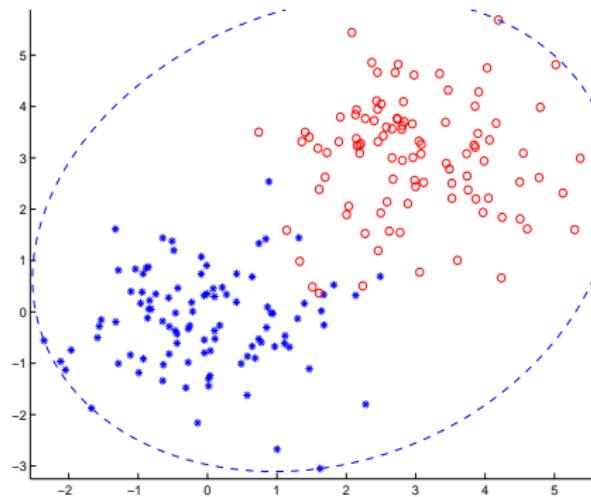
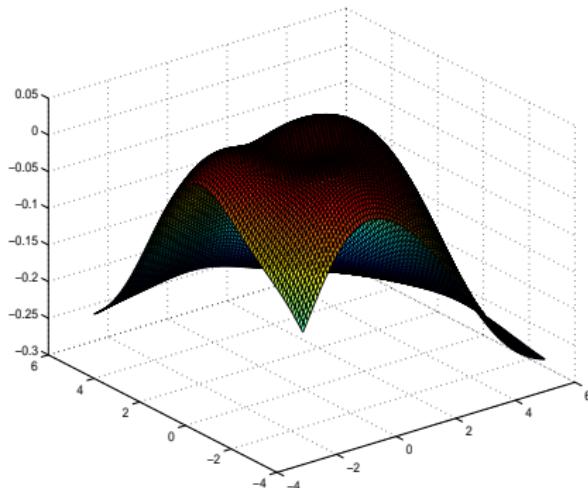


Figure – Ne colle pas suffisamment aux données

## Topo sur la classification

Les noyaux permettent de modéliser efficacement des non-linéarités douces



Figure

## Topo sur la classification

Les fonctions de décision s'écrivent en terme de noyaux :

$$f(x) = \sum_{i=1}^m c_i k(x, x_i) + b \geq 0$$

où

- ▶  $x_i$ ,  $i = 1 \dots m$  sont les vecteurs d'apprentissage
- ▶  $x$  est le vecteur testé
- ▶  $k(\cdot, \cdot)$  est le noyau
- ▶  $c_i$  sont les coefficients d'altitude
- ▶  $-b$  est l'altitude de coupure

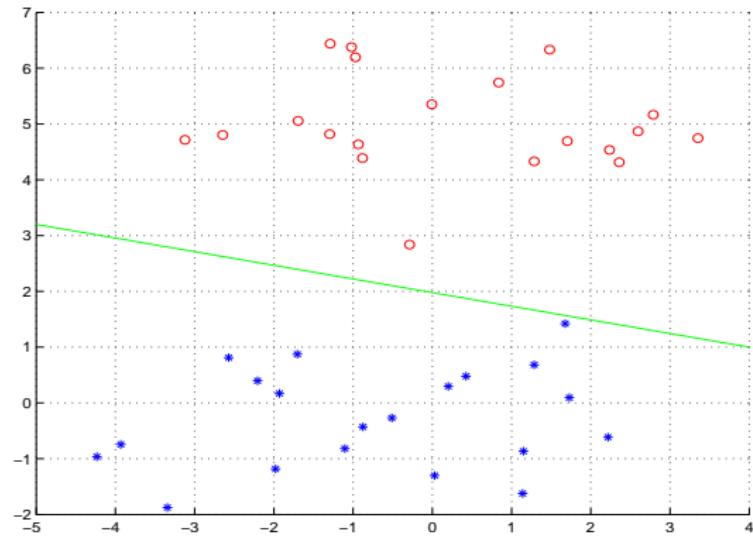
## SVM à deux classes

Les SVM (Support Vector Machines) sont une méthode de classification.

- ▶ on considère un ensemble d'apprentissage  $(x_1, y_1), \dots, (x_m, y_m)$  où  $y_i$  est le label de la classe du vecteur  $x_i$
- ▶ Les vecteurs  $x_i$  prennent leur valeur dans  $\mathcal{X}$
- ▶ Les labels  $y_i$  prennent leur valeur dans  $\mathcal{Y} = \{-1, 1\}$
- ▶ Hypothèse :  $\mathcal{X}$  est muni d'un produit scalaire noté  $\langle \cdot, \cdot \rangle$

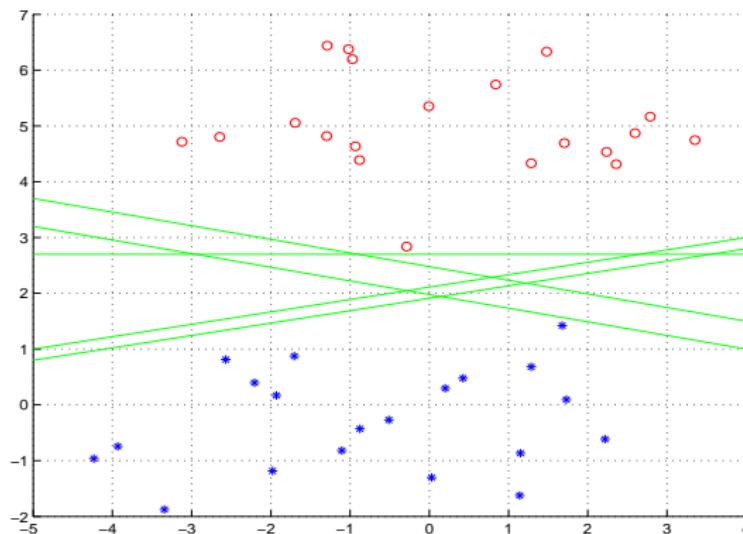
## SVM à deux classes parfaitement séparables

- ▶ On suppose qu'un classifieur linéaire parfaitement séparant peut être mis en œuvre



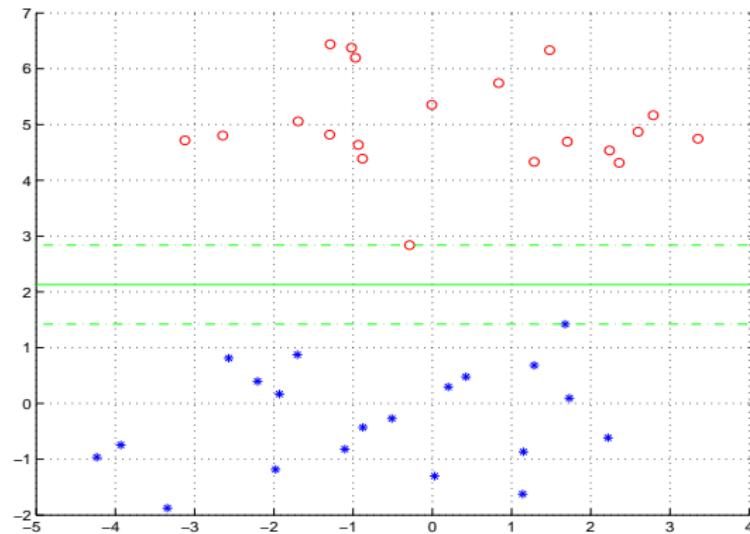
## SVM à deux classes parfaitement séparables

- ▶ Question : quel est le meilleur hyper-plan ???



## SVM à deux classes parfaitement séparables

- ▶ Suggestion : celui qui maximise la marge



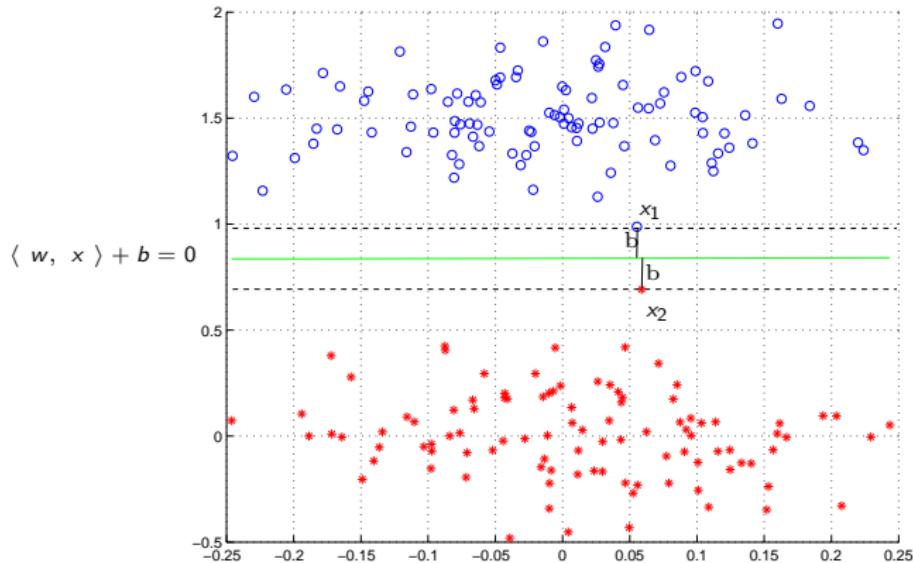
V. Vapnik, S. Kotz, Estimation of Dependences Based on Empirical Data, Springer Series in Statistics, 1982

## SVM à deux classes parfaitement séparables

- ▶ un hyperplan (qui est une droite si  $x_i$  est de dimension 2) est paramétré par  $w$  et  $b$  tel que  $\langle x, w \rangle + b = 0$
- ▶ le vecteur  $w$  est perpendiculaire à l'hyperplan
- ▶ il s'agit de déterminer l'hyperplan optimal : c'est-à-dire en fait tel que  $b$ , qui correspond à la largeur de la marge, soit maximum
- ▶ ce qui donne la fonction de décision :

$$\hat{y} = f(x) = \text{sign}(\langle x, w \rangle + b)$$

## SVM à deux classes parfaitement séparables



$$\begin{aligned} \langle w, x_1 \rangle + b &= 1 \\ \text{et} \\ \langle w, x_2 \rangle + b &= -1 \\ \text{donc} \\ 2b &= \frac{2}{\|w\|} \end{aligned}$$

## SVM à deux classes parfaitement séparables

On a :

- ▶  $b = \frac{1}{\|w\|}$  que l'on veut maximiser

⇒ et qui a le même maximum que  $\frac{1}{\|w\|^2}$  et  $\frac{2}{\|w\|^2}$

- ▶ tout en respectant  $y_i(\langle w, x_i \rangle + b) - 1 \geq 0, \forall i$

## SVM à deux classes parfaitement séparables

Ce problème est un problème d'optimisation non linéaire sous contraintes, d'où son écriture sous forme lagrangienne (somme de la fonction à minimiser et de l'opposé de chaque contrainte multipliée par un multiplicateur de Lagrange  $\alpha_i$ , par définition  $\geq 0$ ) :

$$L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i (\langle w, x_i \rangle + b) - 1)$$

qu'il faut minimiser d'une part par rapport à  $w$  et  $b$  et dont d'autre part il faut annuler les dérivées par rapport aux  $\alpha_i$ .

## SVM à deux classes parfaitement séparables

**Étape 1.** Les minimisations en  $w$  et  $b$  étant effectuées analytiquement, les solutions sont injectées dans  $L_p$

**Étape 2.** et il reste à résoudre numériquement :

**maximiser en**  $\alpha$      $W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$

**avec**                       $\alpha_i \geq 0, i = 1, \dots, m$

**et**                          $\sum_{i=1}^m \alpha_i y_i = 0$

## SVM à deux classes parfaitement séparables

► Finalement, on obtient :

- $w = \sum_{i=1}^m \alpha_i y_i x_i$
- $b = \frac{1}{\|w\|}$
- la fonction de décision ; soit un nouveau point  $x_k$  à classer :

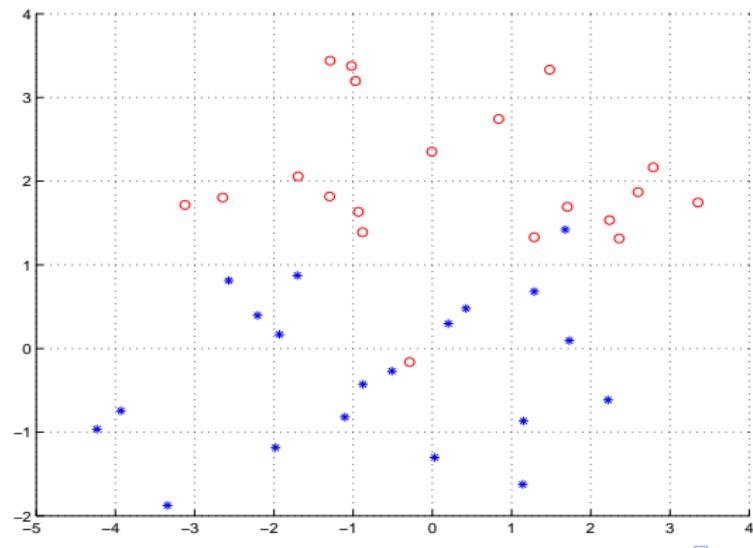
$$y_k = \text{sign} \left( \sum_{i=1}^m y_i \alpha_i \langle x_k, x_i \rangle + b \right)$$

## SVM à deux classes parfaitement séparables

- ▶ Les  $\alpha_i$  sont presque tous égaux à 0 ; les  $x_i$  pour lesquels  $\alpha_i$  est différent de 0 sont appelés les **vecteurs support** ; ces vecteurs sont sur la marge
- ▶ de ce fait aussi, le calcul de la fonction de décision est très rapide

## SVM à deux classes NON parfaitement séparables

Si le classifieur linéaire parfaitement séparant n'existe pas, que faire ? Note : ça correspond aux cas de la réalité !!!



## SVM à deux classes NON parfaitement séparables

Deux solutions :

- ▶ Utiliser le « truc du noyau » : remplacer le produit scalaire  $\langle x, w \rangle$  par un noyau de Mercer  $k(x, w)$ , ce qui permet de passer dans un espace de plus grande dimension, espace où le classifieur linéaire parfaitement séparant existe
- ▶ Et/ou : Autoriser une fraction  $\nu$  d'éléments anormaux (marges douces).

## SVM à deux classes – Le truc du noyau

Pourquoi est-il correct de remplacer  $\langle \cdot, \cdot \rangle$  par  $k(\cdot, \cdot)$  ?

- ▶ certains noyaux, appelés noyaux de Mercer, ont la propriété suivante :  $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$   
(note : dans le petit support de cours, je donne des références pour aller voir ce que ça veut dire)
- ▶ et rien n'interdit de considérer le produit scalaire dans un espace  $(\mathcal{X}, \text{où } \langle x_1, x_2 \rangle)$  ou dans l'autre  $(\mathcal{F}, \text{où } \langle \phi(x_1), \phi(x_2) \rangle)$
- ▶ c'est-à-dire de chercher l'hyperplan séparateur dans l'espace de transformation  $\mathcal{F}$
- ▶ particularité : les  $\langle \phi(x_1), \phi(x_2) \rangle$  ne sont pas directement évalués à partir des  $\phi(x_i)$  puisque ceux-ci ne sont JAMAIS calculés : le truc du noyau permet d'échapper à cela

## SVM à deux classes – Le truc du noyau

Bref, le truc du noyau, c'est :  
remplacer  $\langle \cdot, \cdot \rangle$  par  $k(\cdot, \cdot)$

## SVM à deux classes – Le truc du noyau

La fonction de décision s'écrit alors :

$$y_k = \text{sign} \left( \sum_{i=1}^m y_i \alpha_i k(x_k, x_i) + b \right)$$

Et le problème d'optimisation sous contraintes s'écrit :

**maximiser en**  $\alpha$      $W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j)$

**avec**                       $\alpha_i \geq 0, \quad i = 1, \dots, m$

**et**                          $\sum_{i=1}^m \alpha_i y_i = 0$

## SVM à deux classes – intérêt du truc du noyau

Noyau polynomial, avec  $p = 2$

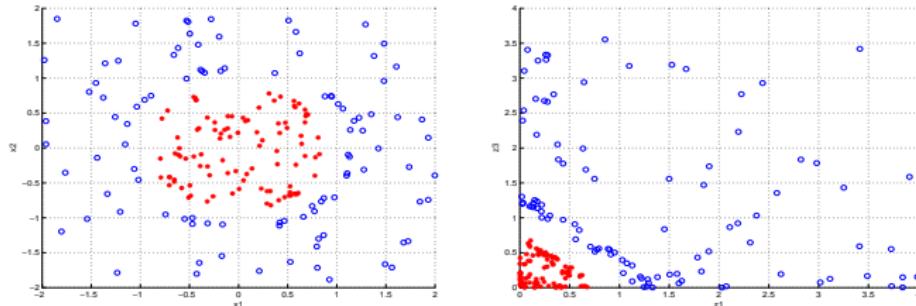


Figure – Gauche : Données brutes. Droite : Données << noyautées >>

- ▶ avec  $z_1 = x_1^2$ ,  $z_2 = \sqrt{2}x_1x_2$  et  $z_3 = x_2^2$
- ▶  $z_2$  est là pour que le noyau soit de Mercer :
 
$$\begin{aligned}\langle \phi(x), \phi(y) \rangle &= (x_1^2, \sqrt{2}x_1x_2, x_2^2), (y_1^2, \sqrt{2}y_1y_2, y_2^2)^T \\ &= ((x_1, x_2), (y_1, y_2)^T)^2 = (\langle x, y \rangle)^2 = k(x, y)\end{aligned}$$

## SVM à deux classes – intérêt du truc du noyau

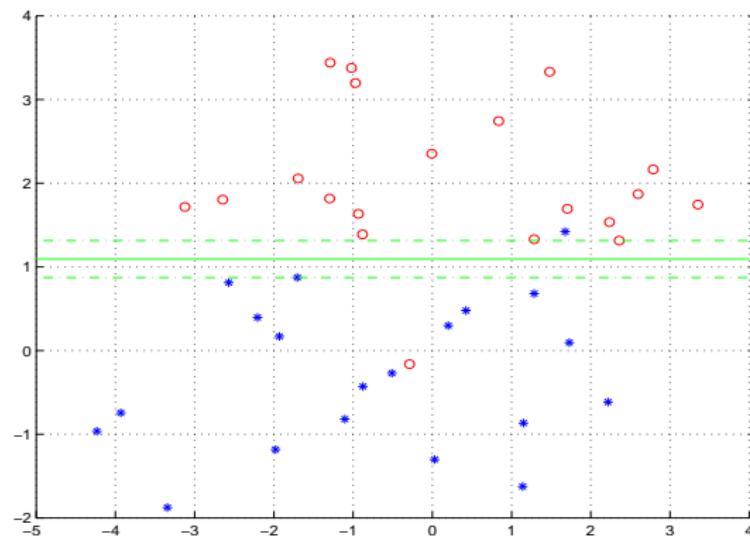
Cette propriété est importante. En effet, considérons l'extraction de tous les monômes de degré  $d$  pour des données de dimension  $N$  (dans l'exemple ci-dessus, l'on a  $d = 2$  et  $N = 2$ ). Si l'on travaille avec des images de dimension  $N = 16 \times 16 = 256$ , et avec  $d = 5$ , on obtient environ  $10^{10} \phi(x)$  à calculer. Le truc du noyau nous permet d'éviter cette explosion combinatoire !

## SVM à deux classes – Marge douce

- ▶ dans les applications réelles, l'ensemble d'apprentissage contient des éléments anormaux
- ▶ note : le truc du noyau peut être considéré aussi ici... ce que je fais directement
- ▶ dans les applications réelles, l'ensemble d'apprentissage contient des éléments anormaux **même dans l'espace de transformation**
- ▶ on souhaite ignorer  $\nu m$  cas anormaux (c'est-à-dire mettre en place des marges douces)

## SVM à deux classes – Marge douce

C'est-à-dire, considérer ceci :



## SVM à deux classes – Marge douce

Pour autoriser des éléments anormaux du mauvais côté de la marge, il faut :

$$\text{minimiser} \quad \frac{1}{2} \|w\|^2 - \nu b + \frac{1}{m} \sum_{i=1}^m \xi_i$$

avec  $y_i k(w, x_i) \geq b - \xi_i$  pour tout  $i = 1, \dots, m$

Les  $\xi_i$  sont des variables de relâchement et  $\nu$  détermine la fraction maximum d'éléments anormaux

## SVM à deux classes – Marge douce

Le problème numérique d'optimisat° sous contraintes s'écrit alors :

**maximiser**     $W(\alpha) = -\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j)$

**avec**               $0 \leq \alpha_i \leq \frac{1}{m}, \quad i = 1, \dots, m$

**et**               $\sum_{i=1}^m \alpha_i y_i = 0$               **et**               $\sum_{i=1}^m \alpha_i \geq \nu$

Fonct° de décision encore :  $y_k = \text{sign} \left( \sum_{i=1}^m y_i \alpha_i k(x_k, x_i) + b \right)$

## Généralisation des SVM classiques à $N$ classes

### 1. Stratégie « Un contre tous ».

$N$  SVM classiques sont entraînés. Les deux classes, pour le classifieur  $j$ , sont :

**première classe** : échantillons de la classe  $j$

**seconde classe** : échantillons de toutes les autres classes

Les paramètres de ces  $N$  classificateurs sont, pour  $j = 1 \dots N$  :  $\alpha_i^j$  pour  $i = 1 \dots m$ ; et  $b^j$ .

La classe d'un nouvel échantillon à classer  $x_k$  est le  $j$  pour lequel  $g^j(x_k)$  est maximum avec :

$$g^j(x_k) = \sum_{i=1}^m y_i \alpha_i^j k(x_k, x_i) + b^j$$

## Généralisation des SVM classiques à $N$ classes

2. Stratégie « Classification paire-par-paire ». Un SVM classique est entraîné pour chaque paire de classes possible. C'est-à-dire que  $n = \frac{(N-1)N}{2}$  SVM classiques sont entraînés. Considérons un nouvel échantillon à classer  $x_k$ . Les sorties des  $n$  classifieurs sont évaluées. La classe de  $x_k$  est celle qui récolte le plus de vote.
3. Stratégie « Error-Correcting Output Coding ». Cette fois, c'est le problème lui-même qui est découpé en sous-problèmes, à chaque fois binaires. Par exemple, si l'on considère le problème de la classification des 10 chiffres (0 à 9), il s'agirait de considérer par exemple la classe des chiffres pairs d'un côté et la classe des chiffres impairs de l'autre ; puis de continuer à diviser chacune de ces deux classes de façon binaire.

## Généralisation des SVM classiques à $N$ classes

4. Stratégie « Multi-Class Objective Functions ». Il y a moyen de réécrire le problème d'optimisation sous contraintes donné ci-dessus dans le cas où plus de deux classes sont à séparer. Il faut réécrire toutes les formules.
5. Une autre stratégie, est de considérer les SVM 1 classe (plutôt que les SVM classiques, c'est-à-dire à deux classes) : voir ci-dessous.

## SVM 1 classe – Introduction

- ▶ Nous avons vu ci-dessus que les SVM, classiquement, permettent de classer des données en deux classes. Pour étendre ces SVM au cas où l'on est en présence d'un problème à plus de deux classes, il faut utiliser un stratagème. Quatre sont donnés dans la section précédente.
- ▶ Mais, ici, nous allons considérer un autre cas : le cas où les données d'apprentissage appartiennent à une seule classe (zone stable/stationnaire d'un signal). Puis on va étendre cette façon de voir les choses quand il y a plus d'une classe (en fait, partir des SVM classiques, à deux classes, n'est pas forcément le plus aisé et le plus efficace).

## SVM 1 classe – Introduction

Du coup, deux utilisations sont envisageables :

- ▶ Dans quelle mesure ces « SVM 1 classe » permettent de déterminer la densité de probabilité correspondant à la classe déterminée lors de l'apprentissage.
- ▶ Dans quelle mesure, quand une nouvelle donnée est présentée à ces « SVM 1 classe », il est possible de déterminer si cet échantillon appartient ou non à la classe déterminée lors de l'apprentissage.

## SVM 1 classe – Introduction

Donc, ce que l'on appelle les « SVM 1 classe » peut-être utilisé (voir les domaines d'application de l'apprentissage supervisé donnés au début) :

- ▶ pour la détection de changements brusques (par exemple, en ce qui concerne la musique, pour la segmentation en notes)
- ▶ ainsi que pour déterminer la frontière/les frontières de fonctions de décision. Il s'agit ici de trouver une région de l'espace des données, qui contient une majorité, réglée par  $\nu$ , des données d'apprentissage.

## SVM 1 classe – Un peu de théorie

Le problème d'optimisation sous contraintes tel qu'il est donné ci-dessus doit être réécrit. La marge à considérer doit bien sûr être douce. On obtient finalement :

**minimiser en**  $\alpha$      $W(\alpha) = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j k(x_i, x_j)$

**avec**                       $0 \leq \alpha_i \leq \frac{1}{\nu m}, \quad i = 1, \dots, m$

**et**                               $\sum_{i=1}^m \alpha_i = 1$

## SVM 1 classe – Un peu de théorie

Le problème est QP<sub>+</sub> (quadratique convexe). En effet, il peut s'écrire :

$$\text{minimiser en } x \quad c^T x + \frac{1}{2} x^T K x$$

avec  $I \leq x \leq u$

et  $Ax = d$

⇒ alors, il existe plein de méthodes numériques pour le résoudre

## SVM 1 classe – Un peu de théorie

en considérant que :

$$c = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}; \quad l = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}; \quad u = \frac{1}{\nu m} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}; \quad x = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_m \end{bmatrix}; \text{ et}$$
$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_m, x_1) \\ \vdots & \ddots & \vdots \\ k(x_1, x_m) & \cdots & k(x_m, x_m) \end{bmatrix}$$

ainsi que :

$$d = 1; \text{ et } A = [1 \dots 1]$$

## SVM 1 classe – Un peu de théorie

Il existe de nombreuses manières pour optimiser numériquement :

- ▶ gradient conjugué avec contraintes
- ▶ méthode de projection
- ▶ décomposition de Bunch-Kayfman
- ▶ méthodes de points intérieurs (le bien connu LOQO)
- ▶ SMO (« Sequential Minimal Optimization ») : dans le petit support de cours que je fournis, SMO est complètement décrit, en détails

## SVM 1 classe – Un peu de théorie

La marge  $b$  est alors calculée de la façon suivante :

$$b = - \sum_{j=1}^m \alpha_j k(x_j, x_i)$$

où  $x_i$  est tout échantillon pour lequel  $\alpha_i$  est différent de 0.

Pour un nouveau point  $x_k$  à classer, l'on considère :

$$y_k = \text{sgn} \left( \sum_{i=1}^m \alpha_i k(x_i, x_k) + b \right)$$

si  $y_k$  est positif, le nouveau point  $x_k$  appartient bien à la même classe que les points qui ont servi à entraîner.

## SVM 1 classe – Exemple

Pour le noyau RBF gaussien vu ci-dessus (slide 189), il n'est pas aisé de séparer les influences respectives de  $2\sigma^2$  et  $\nu$ . Cependant, on peut rapidement indiquer que :

- ▶ Si  $\nu$  est fixé à 0 (pas d'éléments anormaux autorisés), on voit que plus  $2\sigma^2$  est petit plus la frontière colle aux données. Le nombre de nuages augmente, et à la limite s'approche du nombre de données.
- ▶ À  $2\sigma^2$  fixé, si  $\nu$  augmente, le nombre de vecteurs support de marge diminue, puisque certains d'entre eux deviennent des éléments anormaux. De plus, les frontières deviennent plus douces.

⇒ Voir les figures qui suivent !!!

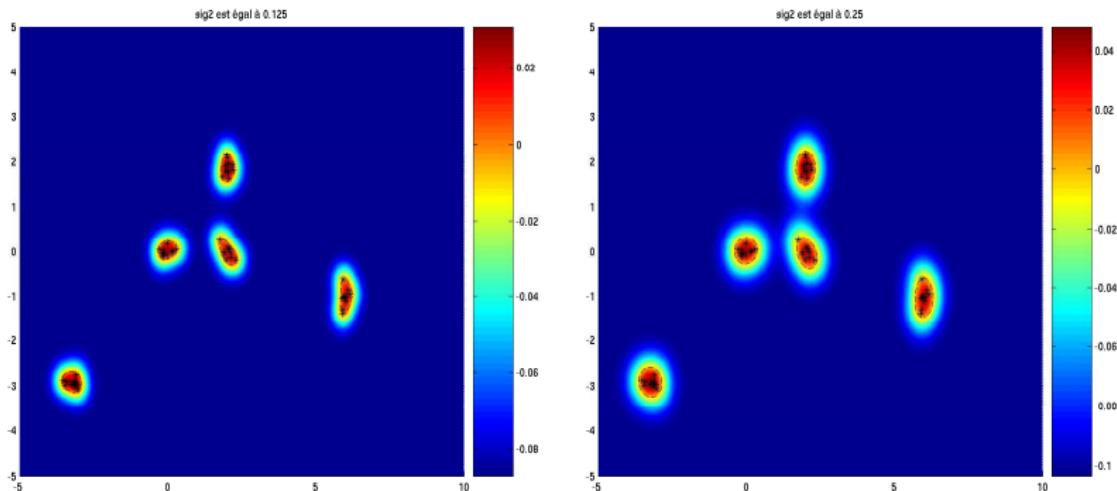


Figure – 5 nuages de points. Noyau : RBF gaussien avec  $2\sigma^2 = 0.125$  et  $2\sigma^2 = 0.25$ .  $\nu = 0.3$ . Étoiles : échantillons. Trait interrompu : fonction de décision.

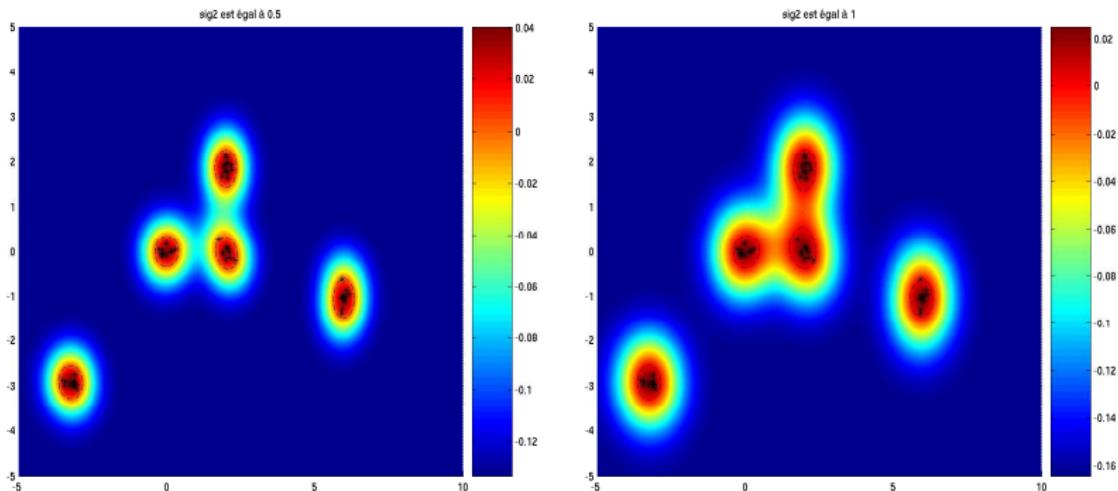


Figure – 5 nuages de points. Noyau : RBF gaussien avec  $2\sigma^2 = 0.5$  et  $2\sigma^2 = 1$ .  $\nu = 0.3$ . Étoiles : échantillons. Trait interrompu : fonction de décision.

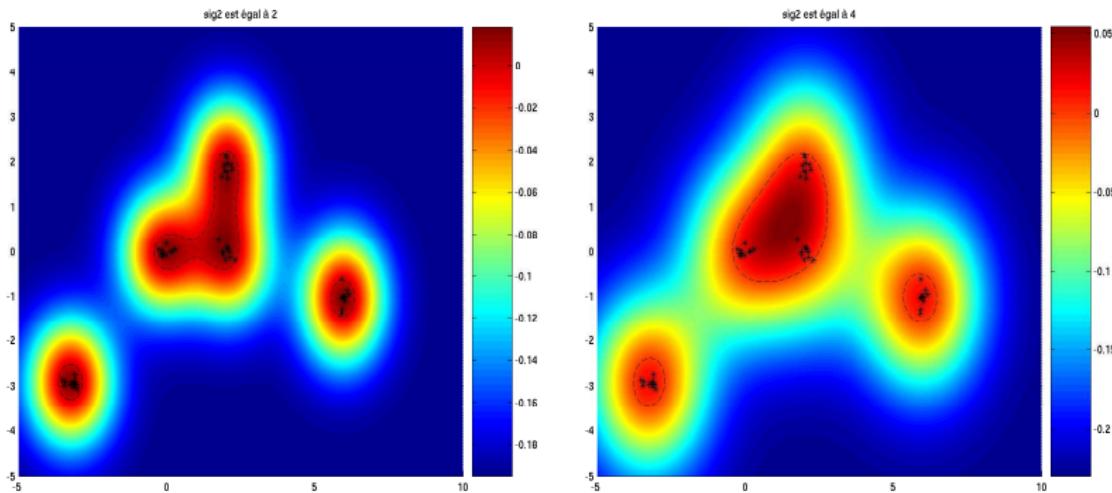
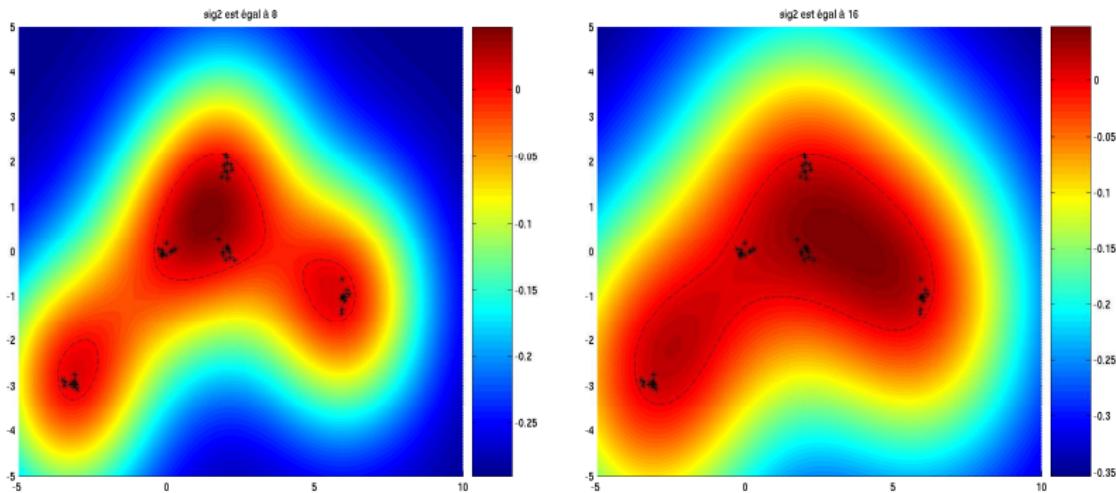


Figure – 5 nuages de points. Noyau : RBF gaussien avec  $2\sigma^2 = 2$  et  $2\sigma^2 = 4$ .  $\nu = 0.3$ . Étoiles : échantillons. Trait interrompu : fonction de décision.



**Figure –** 5 nuages de points. Noyau : RBF gaussien avec  $2\sigma^2 = 8$  et  $2\sigma^2 = 16$ .  $\nu = 0.3$ . Étoiles : échantillons. Trait interrompu : fonction de décision.

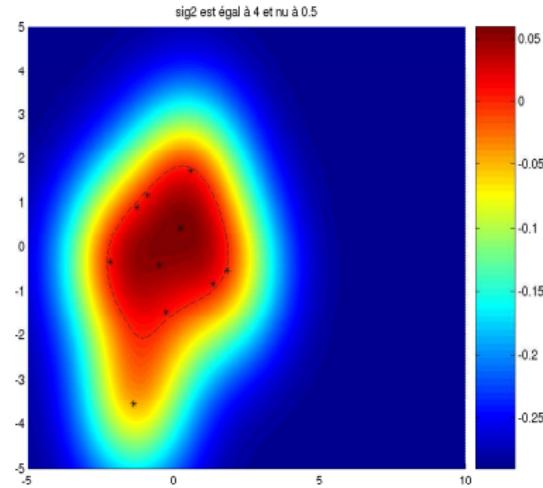
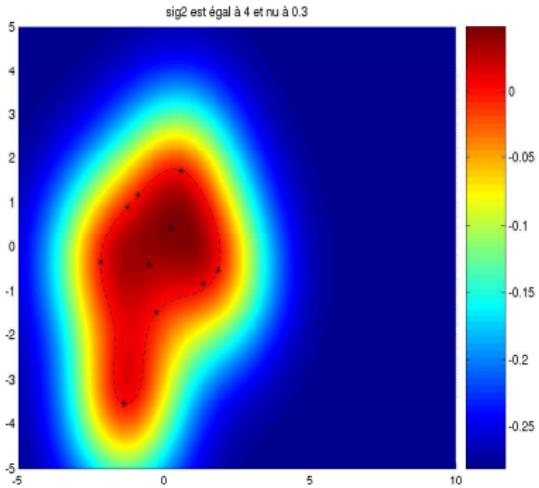


Figure – 1 nuage de points. Le noyau RBF gaussien est utilisé, avec  $2\sigma^2 = 4$ .  $\nu = 0.3$  et  $\nu = 0.5$ . Étoiles : échantillons. Trait interrompu : fonction de décision.

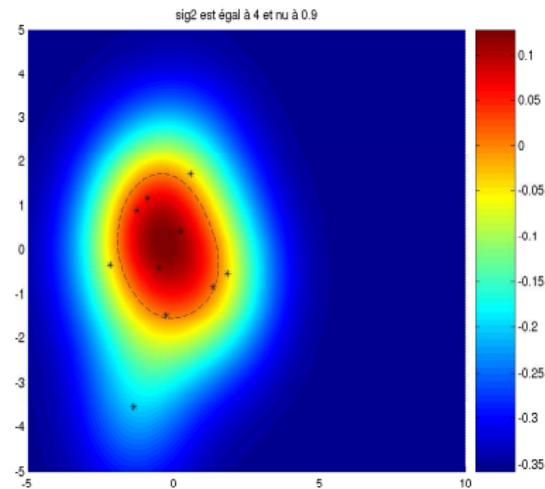
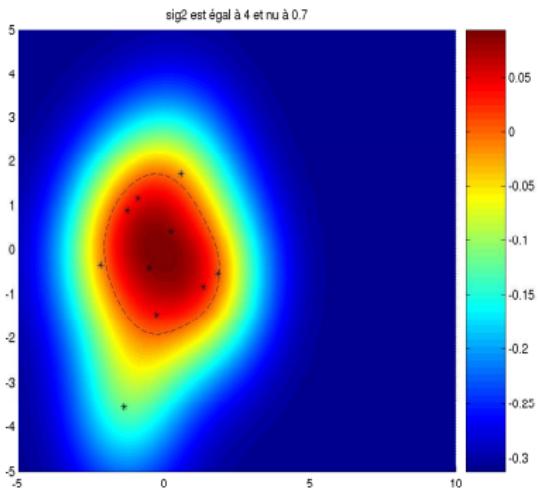


Figure – 1 nuage de points. Le noyau RBF gaussien est utilisé, avec  $2\sigma^2 = 4$ .  $\nu = 0.7$  et  $\nu = 0.9$ . Étoiles : échantillons. Trait interrompu : fonction de décision.

## Quelques noyaux

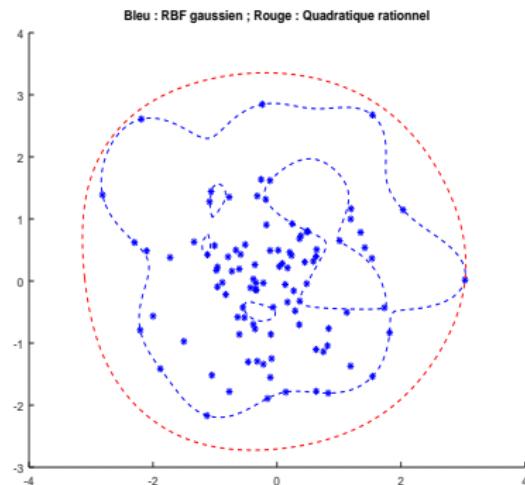


Figure – Un nuage de points gaussiens. Fonctions de décision.

## Application des SVM 1 classe au cas à $N$ classes

- ▶ On a  $N$  classes, et pour chacune d'elles on a calculé un SVM 1 classe
- ▶ On a un nouveau point  $x$ , que l'on cherche à classer
- ▶ Il faut définir des mesures telles que, considérant le point à classer  $x$ , on mesure sa « distance » par rapport à chacune des  $N$  classes
- ▶ Il suffit alors de décider de ranger ce point  $x$  dans la classe pour laquelle on obtient la distance la plus petite

## Définition des trois mesures de dissimilarité – $m_1$

- ▶  $m_1 = -50 \log(< w, x > + b + 1)$
- ▶ avec :
  - ▶  $< w, x > = \sum_{i=1}^m \alpha_i k(x_i, x)$
  - ▶  $b = \frac{1}{||w||}$
  - ▶  $w = \sum_{i=1}^m \alpha_i x_i$
- ▶  $\Rightarrow y = < w, x > + b$  est la fonction de décision classique des SVM 1 classe : si  $y$  est positif, il est décidé que le nouveau point  $x$  à classer appartient à la classe

## Définition des trois mesures de dissimilarité – $m_2$

- ▶ Cette mesure est directement dérivée de la méthode de détection de ruptures qui sera décrite plus tard
- ▶ Sauf que l'on ne compare plus deux SVM 1 classe entre eux, mais on considère que la deuxième trame n'est composée que d'un seul point,  $x$ , c'est-à-dire celui que l'on cherche à classer
- ▶ Il faut réécrire la mesure de dissimilarité définie pour la détection de rupture

## Définition des trois mesures de dissimilarité – $m_2$

Finalement, cette mesure de dissimilarité se réécrit aisément :

- ▶ 
$$m_2 = \frac{2 \min(c_1 p_1)}{\pi} \frac{c_1 x}{c_1 p_1}$$
- ▶ avec :
  - ▶  $c_1 p_1 = \arccos\left(-\frac{b}{\|w\|}\right)$
  - ▶  $c_1 x = \arccos\left(\frac{\langle w, x \rangle}{\|w\|}\right)$
- ▶  $c_1 p_1$  ne fait intervenir que les paramètres d'apprentissage ; aussi, considérant les  $N$  classes, il existe un minimum pour  $c_1 p_1$ , appelé ici  $\min(c_1 p_1)$

## Définition des trois mesures de dissimilarité – $m_3$

- ▶  $m_3 = 100(-\log(< w, x >) + \log(-b) + \text{offset})$
- ▶ avec :  $\text{offset} = -\log(-b_{max})$
- ▶  $b$  ne fait intervenir que les paramètres d'apprentissage ; aussi, considérant les  $N$  classes, il existe un maximum pour les  $b$ , appelé ici  $b_{max}$

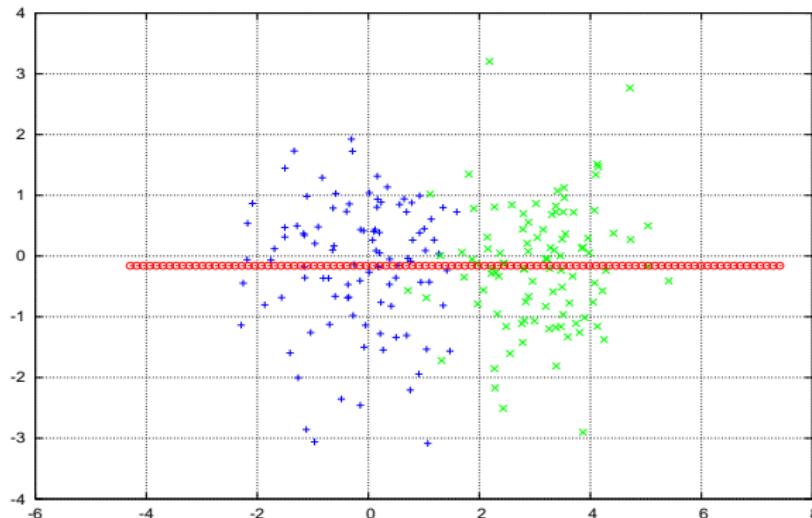
## Comportement des mesures de dissimilarité

- ▶ noyau utilisé :  $k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$  (RBF gaussien)
- ▶ données simulées : 2 classes de données à 27 dimensions, gaussiennes ; puis on fait passer un point à classer à travers ces classes, de gauche à droite, et l'on observe les sorties de  $m_1$ ,  $m_2$  et  $m_3$  de chacun des SVM 1 classe

Note : les 2 classes sont disposées le long de la dimension 1 ; sur cette dimension, la moyenne de la classe 1 est 0 et celle de la classe 2 est 3, et sur la ou les autres dimensions, la moyenne des deux classes est 0 ; les variances sont égales à 1.0.

## Comportement des mesures de dissimilarité – $d = 2$

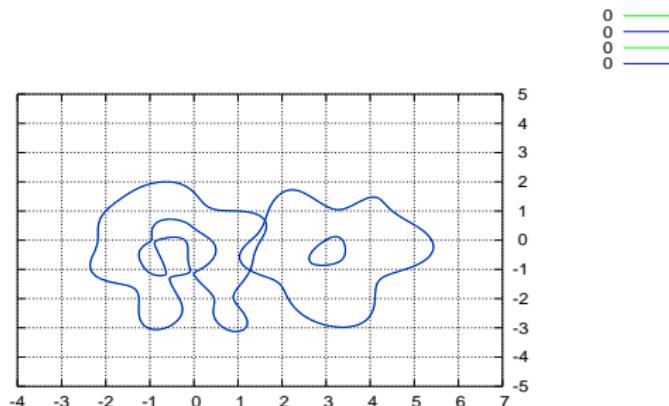
Un exemple de données (en dimension  $d = 2$  !) :



⇒ Importance de la dimensionnalité

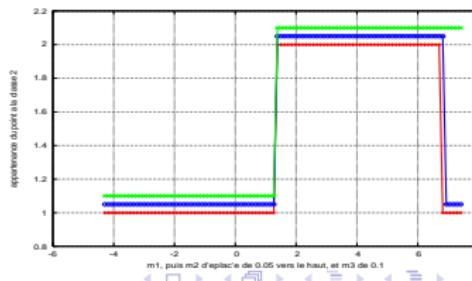
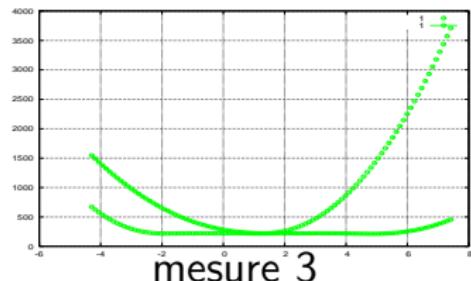
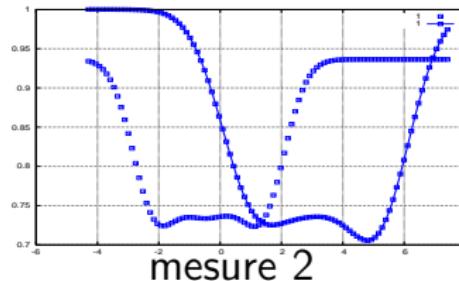
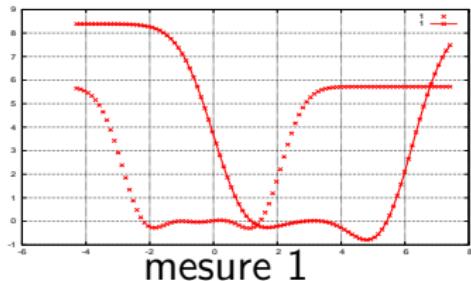
## Comportement des mesures de dissimilarité – $d = 2$

- ▶ classe1 :  $2\sigma^2 = 1.0$  ; classe2 :  $2\sigma^2 = 2.0$
- ▶ frontières trouvées :



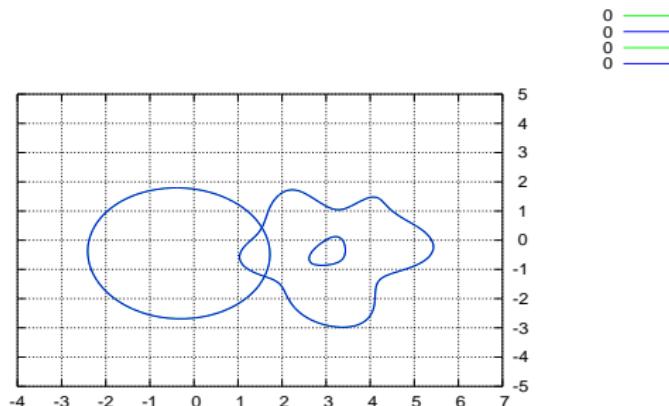
## Comportement des mesures de dissimilarité – $d = 2$

- classe1 :  $2\sigma^2 = 1.0$ ; classe2 :  $2\sigma^2 = 2.0$



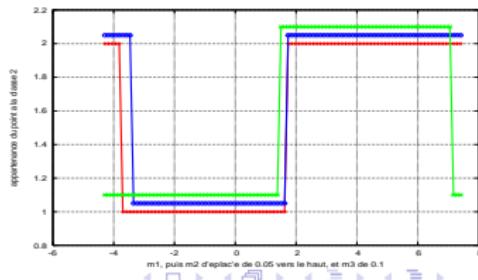
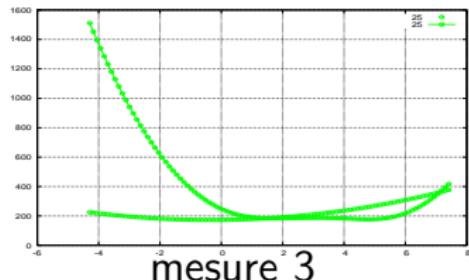
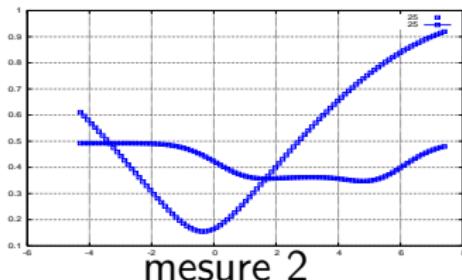
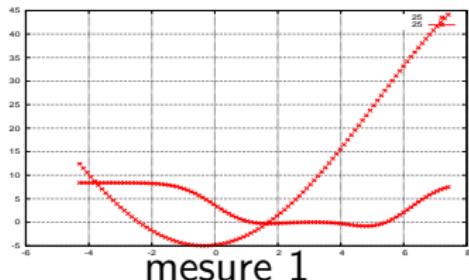
## Comportement des mesures de dissimilarité – $d = 2$

- ▶ classe1 :  $2\sigma^2 = 25.0$ ; classe2 :  $2\sigma^2 = 2.0$
- ▶ frontières trouvées :



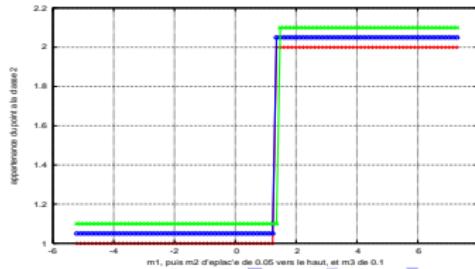
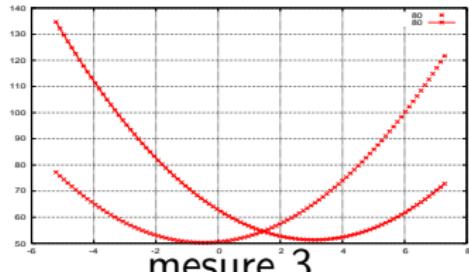
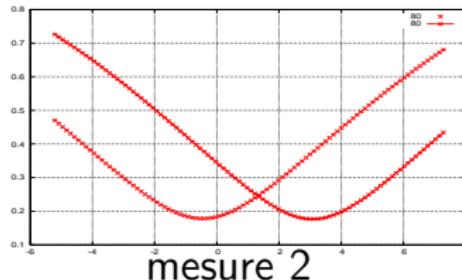
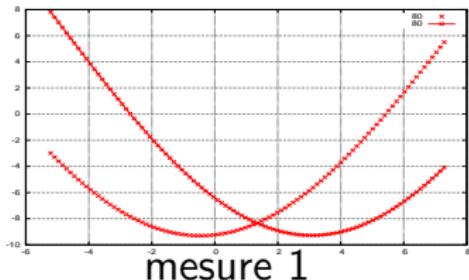
## Comportement des mesures de dissimilarité – $d = 2$

- classe1 :  $2\sigma^2 = 25.0$ ; classe2 :  $2\sigma^2 = 2.0$



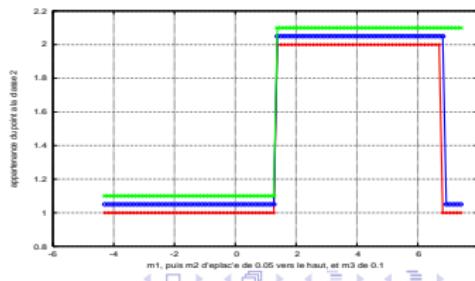
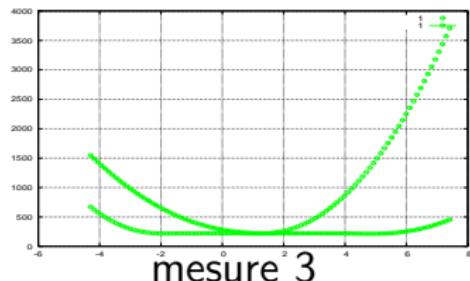
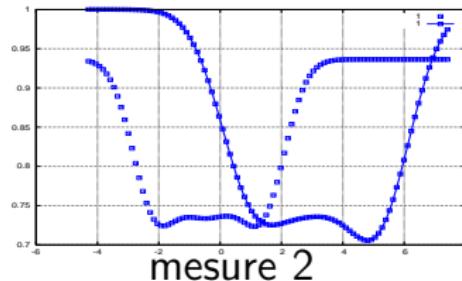
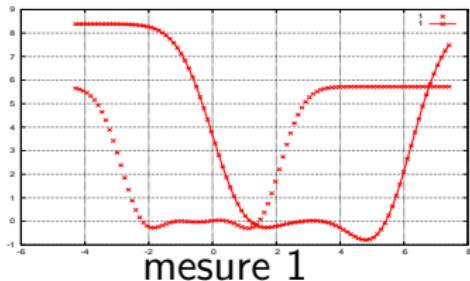
## Comportement des mesures de dissimilarité – $d = 27$

- classe1 :  $2\sigma^2 = 80.0$ ; classe2 :  $2\sigma^2 = 80.0$



## Comportement des mesures de dissimilarité – $d = 27$

- classe1 :  $2\sigma^2 = 80.0$ ; classe2 :  $2\sigma^2 = 1.0$



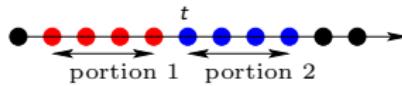
## SVM 1 classe pour la détection de rupture – définitions 1/2

- ▶ **Note : on a besoin de cette partie pour justifier  $m_2$ .**
- ▶ On considère un signal, que l'on sait sujet à passer d'un état stable à un autre, ce via des transitions courtes.
- ▶ Un exemple : un son de flûte enregistré. Chaque zone stable correspond à une note.
- ▶ On veut détecter les transitions entre les notes. Ce, par exemple parce que l'on a pour objectif de faire la transcription automatique du morceau joué, c'est-à-dire, à partir du son, retrouver la partition.
- ▶ Des applications pratiques étant entre autres la détection des fraudeurs, la prédition de pannes (maintenance prédictive)...

## SVM 1 classe pour la détection de rupture – définitions 2/2

L'idée est la suivante :

- ▶ on définit un temps de mesure  $t$
  - ▶ on utilise 2 portions du signal, adjacentes, une précédent et l'autre comprenant et suivant  $t$
  - ▶ on entraîne un SVM 1 classe pour chacune des 2 portions
  - ▶ on mesure la dissimilarité entre les deux SVM 1 classe obtenus : si 1 des portions appartient à une note et l'autre à une 2ième note, la dissimilarité est grande
  - ▶ on fait varier le temps de mesure  $t$  du début à la fin du signal
- ⇒ Les  $t$  pour lesquels la dissimilarité est supérieure à un seuil correspondent chacun à une transition.

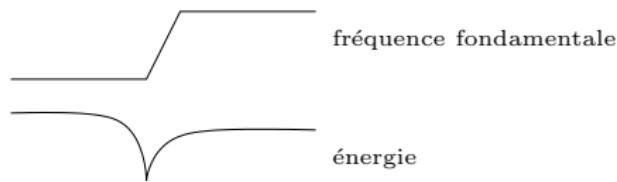


## SVM 1 classe pour la détection de rupture – exemple 1/8

Supposons que l'on extrait les deux caractéristiques suivantes :

- ▶ La fréquence fondamentale du signal enregistré : elle est différente d'une note à l'autre, bien évidemment (remarque : il existe de nombreuses méthodes pour l'extraire, sujet qui ne nous concerne pas ici)
- ▶ L'énergie du signal enregistré : elle diffère d'une note à l'autre (et passe par un minimum au cours de la transition)

Aussi, nous avons par exemple :



## SVM 1 classe pour la détection de rupture – exemple 2/8

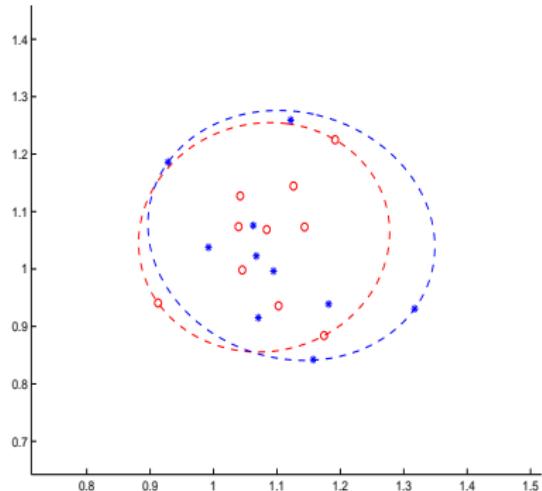
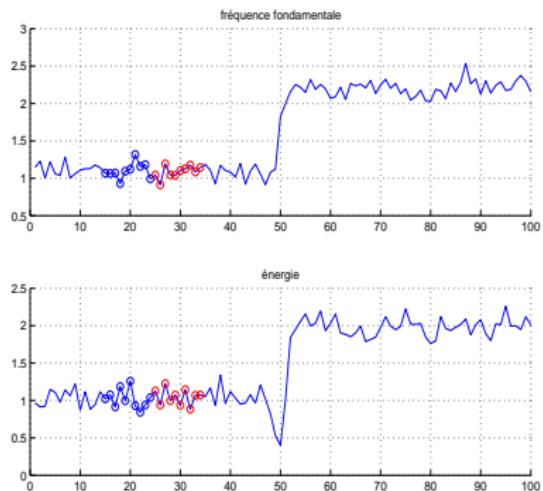


Figure – Bleu : portion 1 ; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 3/8

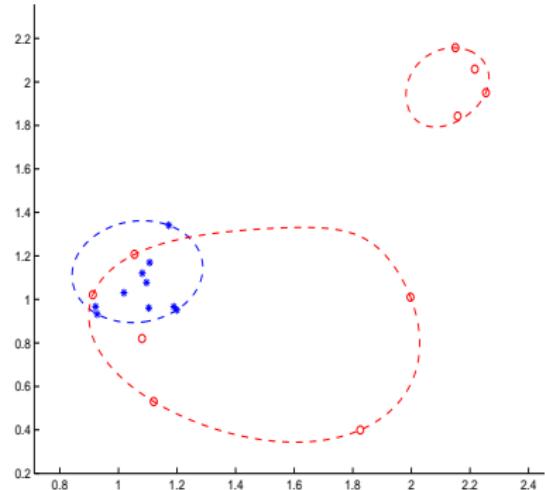
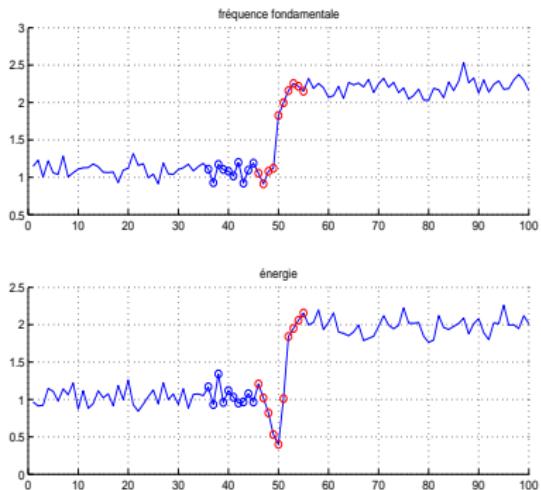


Figure – Bleu : portion 1 ; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 4/8

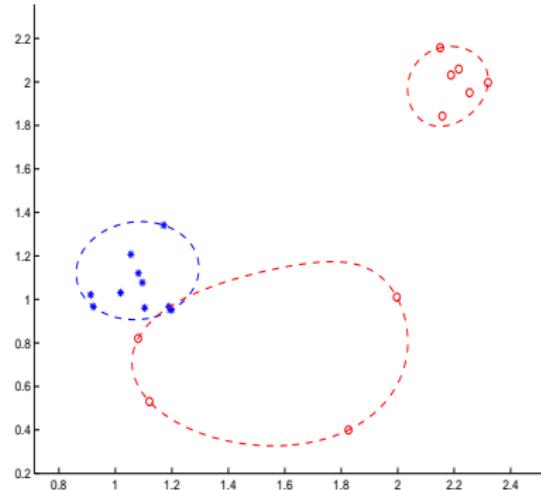
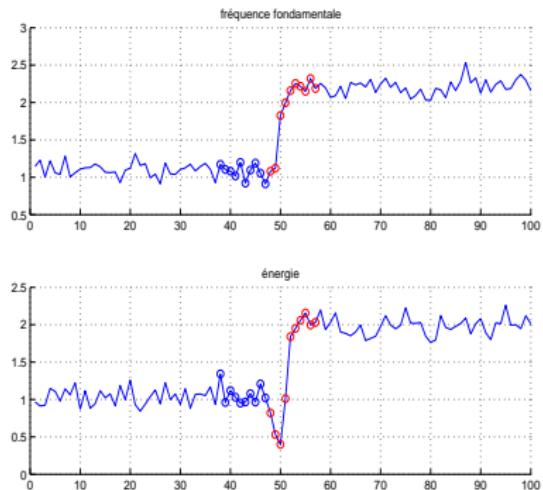


Figure – Bleu : portion 1 ; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 5/8

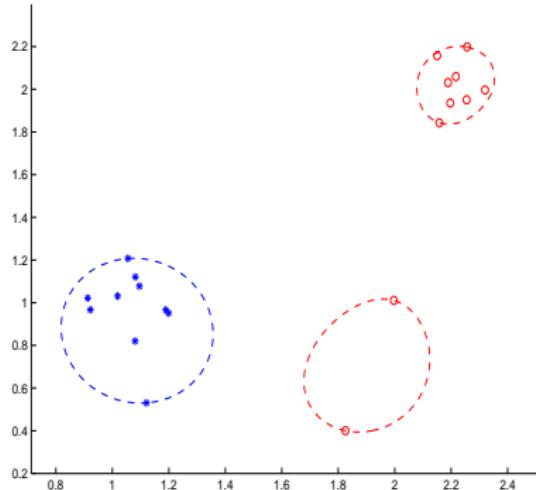
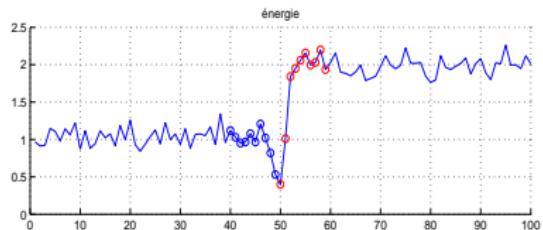
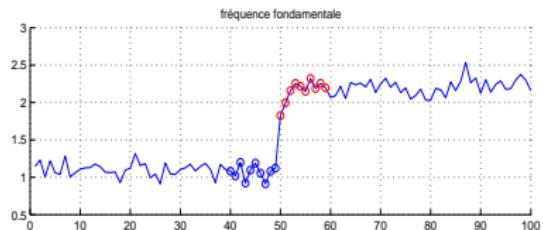


Figure – Bleu : portion 1 ; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 6/8

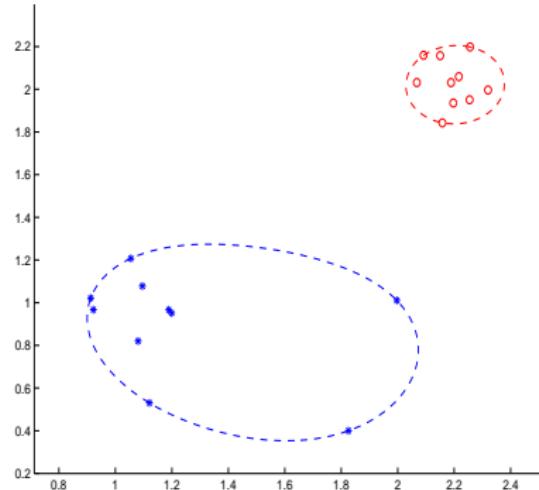
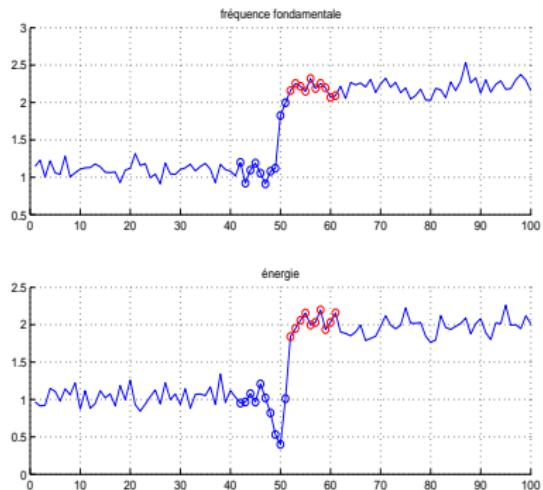


Figure – Bleu : portion 1; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 7/8

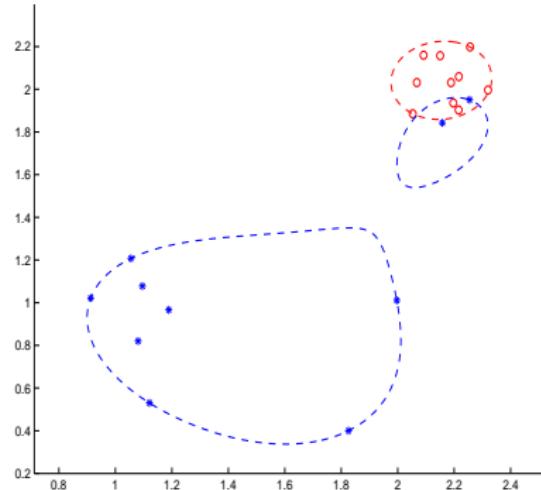
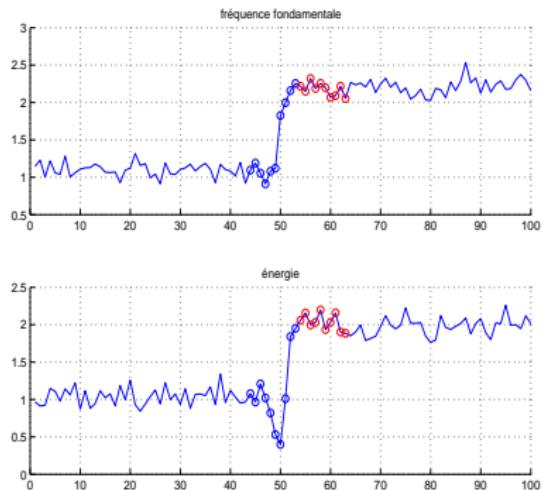


Figure – Bleu : portion 1; Rouge : portion 2

## SVM 1 classe pour la détection de rupture – exemple 8/8

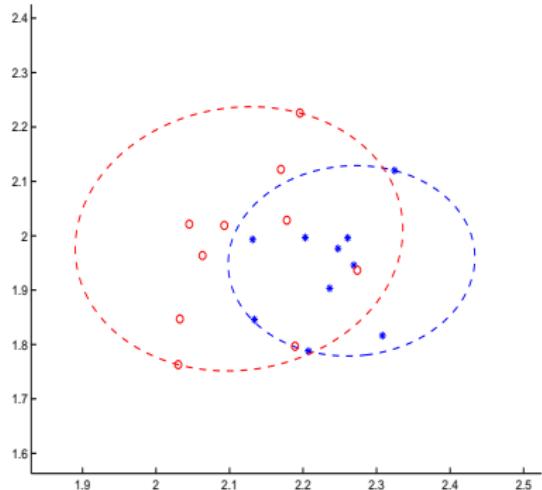
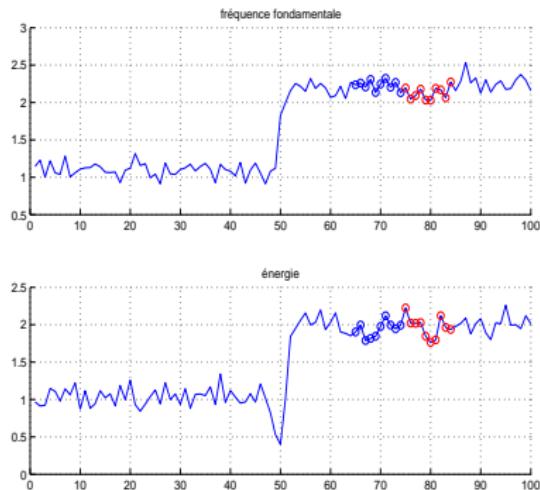


Figure – Bleu : portion 1 ; Rouge : portion 2

## Définition de la mesure de dissimilarité – 1/5

Dans l'espace de transformation, les points après mappage sont tous présents sur une hyper-sphère  $\mathcal{S}$  de rayon 1 centrée à l'origine  $O$  de  $\mathcal{F}$ .

Pourquoi ? Parce que :

$$k(x, x) = \langle \phi(x), \phi(x) \rangle = \|\phi(x)\|$$

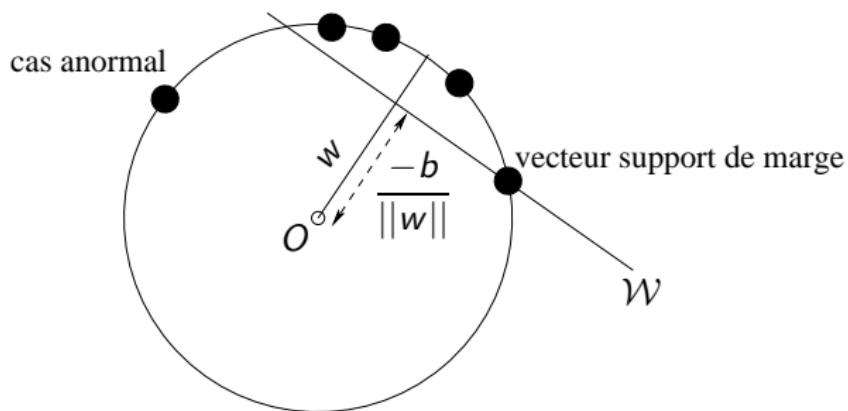
et :

$$k(x, x) = \exp\left(-\frac{(x - x)^2}{2\sigma^2}\right) = 1$$

Donc :  $\|\phi(x)\| = 1$

## Définition de la mesure de dissimilarité – 2/5

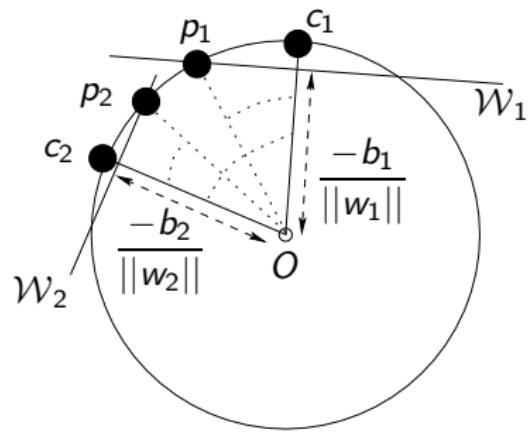
L'hyperplan séparateur  $\mathcal{W}$  est perpendiculaire à  $w$  et sa distance minimale à  $O$  est égale à  $-\frac{b}{\|w\|}$



⇒ Attention : ce mappage n'est jamais explicitement accompli, comme il a déjà été indiqué

## Définition de la mesure de dissimilarité – 3/5

Supposons à présent que nous ayons deux ensembles de données 1 et 2. Pour chacun d'eux, nous entraînons un « SVM 1 classe », séparément. Dans l'espace de transformation, nous obtenons la configuration donnée sur la figure.



## Définition de la mesure de dissimilarité – 4/5

La mesure de dissimilarité est alors :

$$I = \frac{|\widehat{c_1 c_2}|}{|\widehat{c_1 p_1}| + |\widehat{c_2 p_2}|}$$

où  $\widehat{\cdot}$  est l'opérateur « angle entre A et B » ; et où  $p_1$  est le point d'intersection entre  $\mathcal{W}_1$  et l'hypersphère tel que la distance entre  $p_1$  et  $c_2$  soit minimale, et  $p_2$  le point d'intersection entre  $\mathcal{W}_2$  et l'hypersphère tel que la distance entre  $p_2$  et  $c_1$  soit minimale.

## Définition de la mesure de dissimilarité – 5/5

Ceci se réécrit après quelques calculs sous la forme :

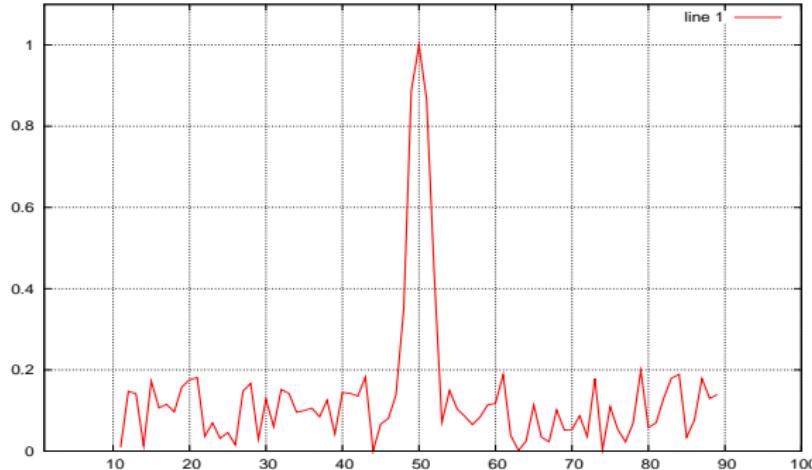
$$(1) \quad \widehat{c_1 c_2} = \arccos \left( \frac{\sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \alpha_{1i} \alpha_{2j} k(x_{1i}, x_{2j})}{|\widehat{c_1 p_1}| + |\widehat{c_2 p_2}|} \right)$$

$$(2) \quad \widehat{c_1 p_1} = \arccos \left( \frac{-b_1}{\left( \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} \alpha_{1i} \alpha_{1j} k(x_{1i}, x_{1j}) \right)^{0.5}} \right)$$

$$(3) \quad \widehat{c_2 p_2} = \arccos \left( \frac{-b_2}{\left( \sum_{i=1}^{m_2} \sum_{j=1}^{m_2} \alpha_{2i} \alpha_{2j} k(x_{2i}, x_{2j}) \right)^{0.5}} \right)$$

## Mesure de dissimilarité et exemple

Comportement de la mesure de dissimilarité pour l'exemple :



## Difficultés rencontrées

- Réglage des paramètres, c'est-à-dire :
  - ▶ le paramètre  $\nu$ , qui contrôle (en partie) le nombre de vecteurs supports
  - ▶ le noyau utilisé
  - ▶ le paramètre  $2\sigma^2$  pour le noyau RBF gaussien : certaines classes peuvent comprendre quelques points, d'autres quelques centaines de milliers ; toutes les classes n'ont pas la même « étendue »
- Problème de mémoire : impossible de former la matrice  $K$  (aussi bien pour la K-PCA que pour les SVM)
- Temps de calcul énorme
- Pour les SVM 1 classe : l'optimiseur SMO (Sequential Minimal Optimization) n'a pas besoin de former la matrice  $K$ , mais en contrepartie est lent

## Estimation du paramètre $2\sigma^2$

- ▶ Critère de Smith :  $\sigma = \left( \prod_{d=1}^N (x_{max}^{(d)} - x_{min}^{(d)}) \right)^{1/N}$
- ▶ Critère de Jaakkola :  
0.5moyenne ou médiane des distances entre tous couples de points  
 $\Rightarrow$  médiane coûteuse :  $(m^2 - 2m + 2)/2$  distances à calculer et stocker

Défaut de Smith :  $2\sigma^2$  augmente avec le nombre de points ; défaut de Jaakkola : le coût en temps de calcul

m	Smith	Jaakkola
100	45.457446	25.120274
1000	80.523818	25.166619
2000	92.347973	25.258566
3000	101.786584	25.145171
8477	120.314888	25.028324

## Réglage du $2\sigma^2$

Avec un  $2\sigma^2$  trop grand, on ne sépare pas assez les classes ; et avec un  $2\sigma^2$  trop petit, on colle trop aux données

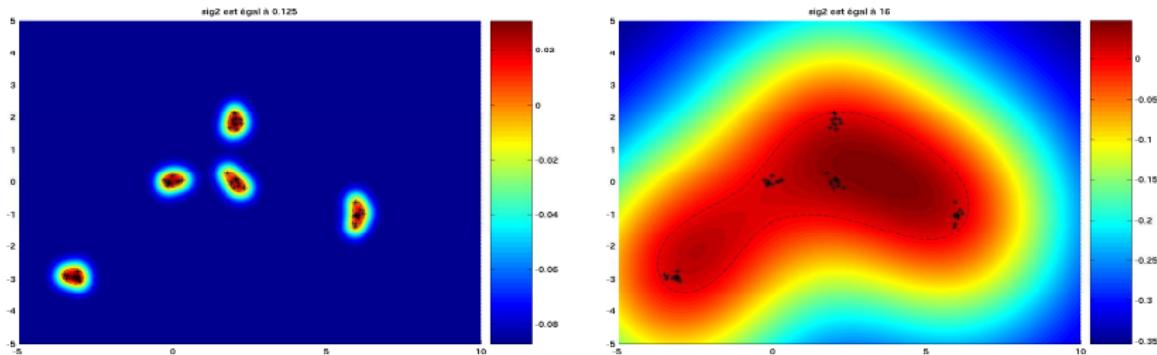


Figure –  $2\sigma^2 = 0.125$  et  $2\sigma^2 = 16$

## Remarques à propos des méthodes à noyaux

- ▶ Pour la K-PCA, la matrice  $K$  est de dimension  $m \times m$  : si  $m$  est très grand, il devient vite impossible de même la former... alors quant à en déterminer les valeurs et vecteurs propres...
  - ▶ De la même façon, en ce qui concerne les SVM, certaines méthodes pour optimiser ont besoin de former  $K$ ... donc si  $m$  est trop grand, là aussi problème
- ⇒ C'est ce qui arrive en parole. Par exemple, on travaille avec des bases de données de plusieurs millions de vecteurs...

## Quelques résultats concernant les SVM 1 classe

- ▶ Donc, les méthodes d'optimisation où il faut former la matrice de Gram  $K$ , comme LOQO, ne sont pas utilisables
  - ▶ pour un problème de reconnaissance de la parole, je me suis retrouvé avec 67 classes comprenant plus de 10000 points
- ▶ La méthode d'optimisation SMO, programmée en *octave*, est beaucoup plus lente que la méthode LOQO

m	LOQO	SMO
150	0.478951	68.101295
250	1.436333	73.216794
400	5.597353	94.421271
1000	70.252285	1286.612408

## Les SVM 1 classe : améliorations/accélérations

- ▶ La méthode d'optimisation SMO, programmée en C sans soin particulier, est encore trop lente
  - ▶ Optimisation des options de compilation
  - ▶ Profiling du code pour déterminer quelles parties du code sont le plus utilisées, et travail sur ces parties du code : par exemple, mise en place de caches
  - ▶ Amélioration de l'algorithme : par exemple, classiquement SMO est initialisé en fixant les  $\nu m$  premiers  $\alpha$ s à des valeurs différentes de 0 ; au lieu de cela, choisir les  $\alpha$ s correspondant aux données les plus éloignées du centre de la classe permet d'énormément accélérer SMO (aussi bien sur des données simulées que sur des données réelles)

v5 : initialisation modifiée ; v7 : cache sur les  $\alpha$ s

m	temps (en s) ; version 4	temps (en s) ; version 5	temps (en s) ; version 7
1000	1.801844	0.064402	0.049241
4000	171.066945	1.271629	0.985059
10000	3984.720679	7.869393	5.992554
20000	69812.153966	29.175334	20.773949
30000	304219.637863	71.555293	59.512090
40000	–	158.614387	132.560738
50000	–	480.471206	422.750329
70000	–	568.750525	485.804473
101000	–	1105.504067	940.501158
200000	–	4212.123195	3717.764977
300000	–	9820.268688	8607.410460
400000	–	17436.723699	15497.636539
500000	–	27109.906805	24407.197990
1000000	–	118816.775029	104593.978641
1500000	–	–	204199.782106
2000000	–	–	378559.661568

## Quelques détails à propos de SMO – Algorithme

L'idée de base : optimiser 2  $\alpha$ s à la fois, ce qui se peut se faire **analytiquement**

Problème : comment choisir ces 2  $\alpha$ s ?

**étape 0.** Une fraction  $\nu$  des  $\alpha_i$ , choisis aléatoirement, sont mis à  $\frac{1}{\nu m}$

**étape 1.** choix du premier  $\alpha$ , d'indice  $i$ . En alternant a) un parcours sur tous les  $\alpha$ , à b) des parcours sur les exemples anormaux ( $\alpha = 1/\nu m$ ), la condition d'arrêt pour b) étant qu'aucun changement n'ait été effectué lors de l'étape 2, et alors on retourne en a)

**Condition d'arrêt** : si aucun changement n'a été effectué en parcourant a), on sort de l'algorithme, l'optimisation est terminée.

## Quelques détails à propos de SMO – Algorithme

**étape 2.** choix du second  $\alpha$ , d'indice  $k$ . Il y a 3 méthodes, essayées successivement :

- ▶ a) si pour  $i$  une condition de KKT est violée, c'est-à-dire pour laquelle on ait :  $(O_i - b) > 0$  ou  $(b - O_i)/(\nu m) - \alpha_i > 0$  (à  $\epsilon$  prêt) avec  $O_i = \sum_{j=1}^m \alpha_j k(x_j, x_i)$ , on choisit la 2ième variable  $\alpha$ , d'indice noté  $k$ , pour laquelle on a  $|O_i - O_k| \max$ ; on optimise  $\alpha_i$  et  $\alpha_k$  et si l'amélioration est sensible (par ça on entend une modification significative des 2  $\alpha$ s) on retourne en 1. (et 1 changement a été effectué)
- ▶ b) sinon, on inspecte tous les exemples anormaux, à partir d'une position aléatoire; puis on retourne en 1. (si au moins pour un d'eux l'amélioration est sensible, au moins 1 changement a été effectué)
- ▶ c) sinon, on inspecte tous les exemples, à partir d'une position aléatoire, puis on retourne en 1. (aucun ou au moins 1 changement ayant été effectué)

## SVM – Remarque : données numériques ou pas

- ▶ Les noyaux sont des mesures de similarité.
- ▶ On peut donc construire des noyaux sur tous les objets comparables.
- ▶ Les données de l'espace  $\mathcal{X}$  peuvent être très hétérogènes : continues, discrètes, logiques, non-numériques, avec des échelles différentes, etc.
- ▶ Attention : si les données sont hétérogènes, il faut fusionner les différentes sortes de données en respectant leur importance relative.

## SVM – Remarque : données numériques ou pas

**Exemple 1 :**  $x = [x_a, x_b]$  où  $x_a$  est une distance entre villes exprimée en kilomètres et  $x_b$  est une durée de voyage exprimée en heures. Prendre

$$k(x, x') = \exp\left(-\frac{(x_a - x'_a)^2 + (x_b - x'_b)^2}{2\sigma^2}\right)$$

conduit à sur-considérer l'influence des distances (ordre  $10^4$ ) par rapport à celle des durées (ordre 10)

## SVM – Remarque : données numériques ou pas

Rééquilibrer :

$$k(x, x') = \exp\left(-\frac{10^{-8}(x_a - x'_a)^2 + 10^{-2}(x_b - x'_b)^2}{2\sigma^2}\right)$$

Ou distance de Mahalanobis :

$$k(x_i, x_j) = \exp\left(-\frac{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}{2\sigma^2}\right)$$

## SVM – Remarque : données numériques ou pas

**Exemple 2 :** Lorsque  $\mathcal{X}$  est un espace de textes (ou plus généralement de chaînes de caractères, d'acides aminés, etc.), on peut définir un noyau.

Ce en explicitement évaluant les  $\phi(x)$  où  $x$  est une chaîne de caractères.

Le noyau dit « sac de mots » consiste à sélectionner préalablement un ensemble de mots (c'est-à-dire de petites chaînes de caractères)  $b_1 \dots b_n$  et définir :

$$\phi_{b_i}(x) = \beta_i \times \text{nombre occurrences de } b_i \text{ dans } x$$

où chaque poids  $\beta_i$  caractérise l'importance du mot  $b_i$

## SVM – Remarque : données numériques ou pas

Alors le noyau est défini par :

$$k(x, x') = \sum_{i=1}^n \phi_{b_i}(x)\phi_{b_i}(x')$$

Dans une base de données de type commercial ou de type production, il est aisément de définir par avance l'ensemble de mots  $b_i$  qu'on va trouver

## Classification – Méthodes

- Pour chaque signal à traiter, segmenté en trames (exemple : pour les sons, 30 ms) :

- ▶ apprentissage avec  $\underbrace{80\%}_{\text{données d'apprentissage}}$  des segments
- ▶ tests avec les  $\underbrace{20\%}_{\text{données de test}}$  restant

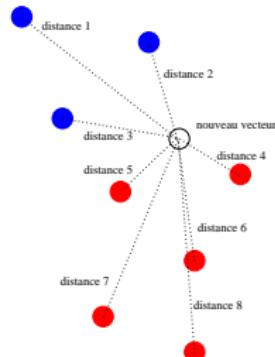
- Méthodes de classification à comparer :

- ▶ Les  $k$  plus proches voisins ( $k$ ppv)
- ▶ Les Mélanges de Gaussiennes (MG)
- ▶ Les Réseaux de Neurones (RN)
- ▶ ⇒ Et ajouter les SVM 1 classe (ou autre)

## Classification – Les $k$ plus proches voisins

- ▶ Pour chaque point à classifier, nous cherchons ses  $k$  plus proches voisins dans le groupe d'apprentissage.
- ▶ Si la majorité de ces  $k$  points appartient à la classe  $i$ , il est décidé que le point à classifier appartient à la classe  $i$ .
- ▶  $k$  est un facteur libre, à fixer.

## Classification – Nouveau point à classer



- ▶ Groupe d'apprentissage : 8 points (3 bleus et 5 rouges)  
**(apprentissage fini)**
- ▶ Choix :  $k = 3$
- ▶ 3 plus petites distances : distances 2 (classe bleue) ; 4, 5 (classe rouge). Donc le nouveau point est rouge.

## Classification – Les $k$ plus proches voisins

- ▶ Ici, nous n'obtenons pas de **paramètres** décrivant (modélisant, compressant) la base de données d'apprentissage (pas de risque empirique), paramètres que nous supposons applicables ensuite pour la base de test (**test 1**, ou risque espéré), puis pour tout nouveau signal à classifier (**test 2**, ou validation croisée). Les données d'apprentissage elles-mêmes constituent les paramètres !!!!
- ▶ Note : la distance communément mise en place est la distance euclidienne : d'autres sont possibles.
- ▶ Défaut de la méthode : sa lenteur.

## Classification – Les mélanges de gaussiennes

- ▶ Les données de chaque classe sont modélisées par une somme de  $P$  densités de probabilité  $p(x|\theta_\nu)$  multi-dimensionnelles gaussiennes.
- ▶ La somme des densités de probabilité  $p(x|\theta_\nu)$  de chaque composante nous donne la densité de probabilité  $P(x|\Theta)$  du mélange pour la classe considérée :

$$P(x|\Theta) = \sum_{\nu=1}^P p(x|\theta_\nu)\pi_\nu$$

## Classification – Les mélanges de gaussiennes

$$P(x|\Theta) = \sum_{\nu=1}^P p(x|\theta_\nu)\pi_\nu$$

- ▶ Avec :  $x$  le vecteur des données (de taille  $m$ ),
- ▶  $\pi_\nu$  les probabilités a priori de chaque composante,
- ▶  $\Theta = (\theta_1 \dots \theta_P)$  et  $\theta_\nu = (y_\nu, \Sigma_\nu)$ ,
- ▶ où  $y_\nu$  est le vecteur des moyennes (sa taille est le nombre  $N$  de dimensions, c'est-à-dire le nombre de caractéristiques prises en compte) pour la composante  $\nu$  et  $\Sigma_\nu$  est la matrice de covariance (de taille  $N \times N$ ) pour la même composante.

## Classification – Les mélanges de gaussiennes

- ▶ Il s'agit de déterminer les **paramètres**  $\Theta$  et  $\pi$  du mélange.
- ▶ La méthode utilisée ici est la méthode du maximum de vraisemblance.
- ▶ Plus communément, nous maximisons suivant  $\Theta$  ceci :

$$\mathcal{L}(\Theta) = \sum_{i=1}^m \log_e P(x_i|\Theta) = \sum_{i=1}^m \log_e \left( \sum_{\nu=1}^P p(x_i|\theta_\nu) \pi_\nu \right)$$

- ▶ Cette maximisation se fait en utilisant l'algorithme EM (« Expectation – Maximization »).

## Classification – Les mélanges de gaussiennes

Les paramètres  $\Theta$  et  $\pi$  sont dans un premier temps initialisés, puis les deux étapes E et M sont répétées jusqu'à la convergence.

Note : ici,  $P$  est supposé connu.

- **Étape E.** Les taux d'assignement  $M_{i\alpha}$  de chaque point  $i$  à chaque composante  $\alpha$  sont estimés (chaque  $x_i$  est lui-même un vecteur de taille  $N$ ) ainsi :

$$\begin{aligned}M_{i\alpha} &= \frac{p(x_i | y_\alpha, \Sigma_\alpha) \pi_\alpha}{\sum_{\nu=1}^P p(x_i | y_\nu, \Sigma_\nu) \pi_\nu} \\&= \frac{|\Sigma_\alpha|^{-1/2} \exp\left(-\frac{1}{2}(x_i - y_\alpha)^T (\Sigma_\alpha)^{-1} (x_i - y_\alpha)\right)}{\sum_{\nu=1}^P |\Sigma_\nu|^{-1/2} \exp\left(-\frac{1}{2}(x_i - y_\nu)^T (\Sigma_\nu)^{-1} (x_i - y_\nu)\right)}\end{aligned}$$



## Classification – Les mélanges de gaussiennes

- **Étape M.** Les paramètres du mélange sont réévalués ainsi :

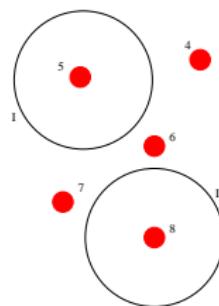
$$y_\alpha = \frac{\sum_{i=1}^m M_{i\alpha} x_i}{\sum_{i=1}^m M_{i\alpha}}$$

$$\Sigma_\alpha = \frac{1}{\sum_{i=1}^m M_{i\alpha}} \sum_{i=1}^m M_{i\alpha} (x_i - y_\alpha)(x_i - y_\alpha)^T$$

$$\pi_\alpha = \frac{1}{m} \sum_{i=1}^m M_{i\alpha}$$

## Classification – Apprentissage GMM

Initialisation. Deux mélanges. Un centré sur 5, l'autre sur 8.  
Co-variances petites.



## Classification – Apprentissage GMM

### Étape E. (taux d'assignments).

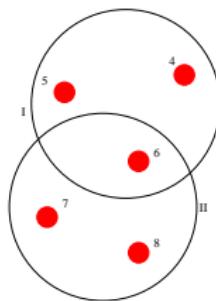
- ▶ 7 appartient plutôt à II.
- ▶ 4 à I.
- ▶ 6 est indéterminé.

### Étape M.

- ▶ La moyenne de II est tirée vers 7 (et aussi vers 6).
- ▶ Celle de I vers 4 (et aussi vers 6).
- ▶ Les co-variances sont augmentées

## Classification – Apprentissage GMM

Après une itération :



... etc. Et ça pour toutes les classes.

## Classification – Nouveau point à classer

On considère le même exemple que pour les  $k$  plus proches voisins (2 classes). Calcul des 2 :

$$P(x|\Theta) = \sum_{\nu=1}^P p(x|\theta_\nu)\pi_\nu$$

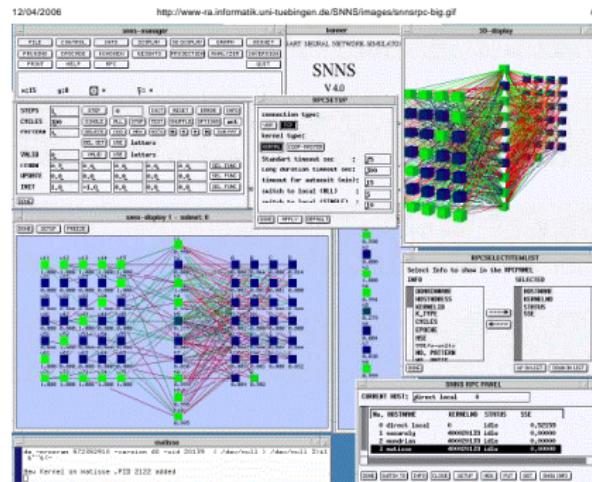
Le nouveau point est rangé dans la classe qui nous donne le plus grand  $P$ .

## Classification – Les réseaux de neurones

- ▶ Pas possible ici de donner des détails sur les réseaux de neurones
- ▶ Références, logiciels (et documentations logiciels) nombreux
- ▶ Mais ça va être trop long à mettre en place dans le cadre de votre TL
- ▶ Je vais présenter des résultats obtenus pour une tâche où les MLP suffisent
- ▶ Mais, bien sûr, il faudrait regarder ce qu'apportent les LSTM, TDNN, GRNN, CNN, et autres (on trouve aisément une vingtaine d'architectures différentes)

# Classification – Les réseaux de neurones

Exemple (simple) d'architecture :



## La tâche : segmentation en sources

**Objectif :** il s'agit de discriminer

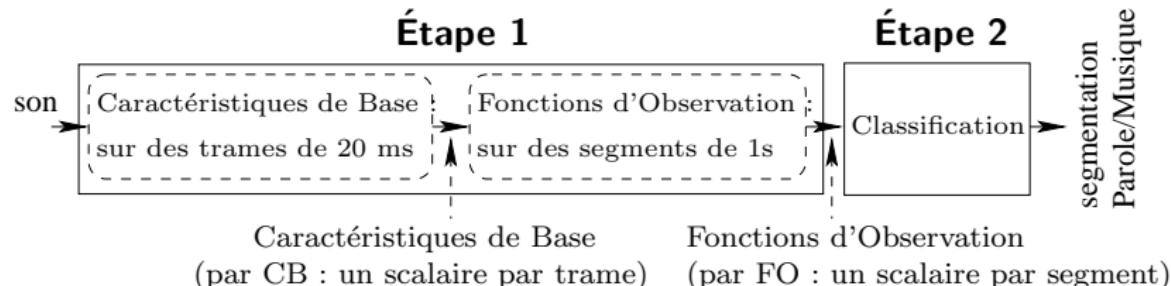
- les **segments** (temporels) de **musique**  
⇒ musique = musique instrumentale et/ou voix chantée
- des **segments** (temporels) de **voix parlée**

**Il faut mettre en évidence :**

- **musique** : succession de zones de stabilité relative
- **voix parlée** : rapide succession de zones de stabilité relative et de zones de bruit

## La tâche – Stratégie

### Procédure :



### Les sons utilisés (pour mes tests) :

- 2 sons enregistrés à la radio longs d'environ 20 minutes chacun :
  - ⇒  $\simeq$  un fichier de 10 minutes de **musique** et un fichier de 10 minutes de **voix parlée**

**Note :** je ne vais entrer dans plus de détails ici (voir le TL)

## Performances de sources

Caractéristique	apprentissage	test 1	test 2
flux spectral	4,7 %	5,4 %	9,1 %
centroïde spectral	7,3 %	8,7 %	12,7 %
taux de passage par zéro	7,4 %	8,3 %	13,2 %
cepstre	8,2 %	10,6 %	13,8 %

- ▶ Fonctions d'observation : *moy* et  $\log_{10}$  de la variance, pour chaque Caractéristique
- ▶ Classifieur utilisé : 7ppv

## Performances de *sources*

Fonction d'observation	test 1
moyenne	37,4 %
$\log_{10}$ de la variance	10,5 %
nombre de trames au-dessous de la moyenne	26,5 %
position $x(i)$ du maximum de l'histogramme	18,2 %
produit $x(i)$ par $\log_{10}$ de la variance du mode	26,8 %

- ▶ Qualité des Fonctions d'observation (caractéristique utilisée : flux spectral)
- ▶ Classifieur utilisé : 7ppv

## Performances de sources

Classifieur	apprentissage	test 1	test 2
MG	4,0 %	3,6 %	9,3 %
kppv ( $k = 7$ )	1,3 %	1,5 %	5,9 %
RN (MLP)	1,9 %	1,7 %	4,7 %

Performances des 3 classifieurs (avec 6 Fonctions d'observation :  
*moy* et  $\log_{10} \text{var}$  du flux spectral, du centroïde spectral et du taux  
de passage par zéro)

## Les erreurs qu'il faut que vous considériez, donc

- ▶ Erreur d'apprentissage (ou risque empirique) : elle est mesurée par le pourcentage d'erreurs de classement (observations mal classées) sur les données d'apprentissage
- ▶ Erreur de généralisation (ou risque espéré, ou réel) : elle est mesurée par le pourcentage d'erreurs de classement (observations mal classées) sur les données de test (qui n'ont pas été utilisées pour faire l'apprentissage)
- ▶ Erreur de validation croisée : elle est mesurée par le pourcentage d'erreurs de classement sur des données "indépendantes" de celles d'apprentissage et de celles de test

## TL – Partie 3a : reconnaissance de chants d'oiseaux

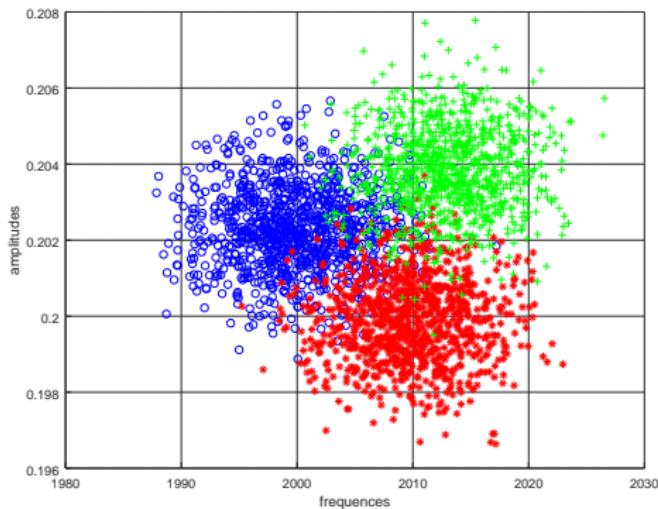
- ▶ Je donne des sons correspondant aux chants de trois oiseaux différents ⇒ il s'agit de les classer automatiquement
- ▶ Les chants sont simples : l'émission d'un signal harmonique sans modulations, par tranches de 50 ms
- ▶ La fréquence de la fondamentale, pour un oiseau donné, n'a pas toujours exactement la même valeur : il y a une certaine dispersion (petite) autour d'une valeur moyenne ; même chose pour les amplitudes
- ▶ Ce qui varie entre les trois chants, c'est la fréquence moyenne de la fondamentale et son amplitude moyenne
- ▶ Cependant, les trois fréquences moyennes et les trois amplitudes moyennes sont proches l'une de l'autre

## TL – Partie 3a : reconnaissance de chants d'oiseaux

- ▶ Lisez les sons avec “audiowrite” (avec une boucle “for”)
- ▶ Extrayez deux caractéristiques sur chacune de ces tranches, avec les méthodes d'analyse spectrale que vous connaissez : la fréquence, et l'amplitude de la fondamentale
- ▶ Possiblement, vous pouvez régler certains paramètres : taille des trames, taille des FFT, fenêtre de pondération utilisée, etc.
- ▶ Je vais aussi vous donner le code permettant de calculer les SVM (1 classe, 2 classes, etc.)
- ▶ Vous aurez aussi à régler les paramètres de ce classifieur
- ▶ Et le but est de me fournir des tableaux du type de ceux présentés juste au-dessus : c'est ça qui m'intéresse

## TL – Partie 3a : reconnaissance de chants d'oiseaux

Vous devriez obtenir des clusters ayant ces allures :



## TL – Partie 3b : segmentation en sources

- ▶ Dans les slides qui suivent, je donne des détails sur les Caractéristiques utilisées, les Fonctions d'observation, etc.
- ▶ Vous n'avez pas à les programmer : je vous donne le code qui permet de les extraire
- ▶ Possiblement, vous pouvez régler certains paramètres : taille des trames, taille des FFT, fenêtre de pondération utilisée, taille des segments
- ▶ Je vais aussi vous donner le code permettant de calculer les SVM
- ▶ Vous aurez aussi à régler les paramètres de ce classifieur
- ▶ Et le but est de me fournir des tableaux du type de ceux présentés juste au-dessus : c'est ça qui m'intéresse

## TL – Partie 3b : Caractéristiques

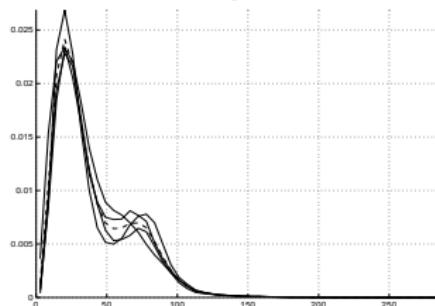
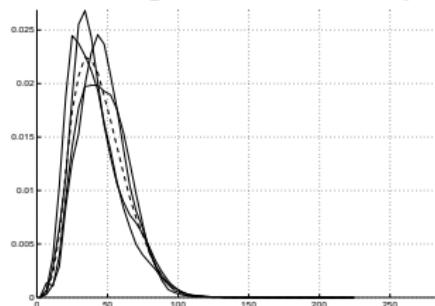
### Caractéristiques mises en place :

- Le flux spectral :  $\sum_i |\hat{S}_1[i]| - |\hat{S}_2[i]|$
- Le centroïde : centre de gravité du spectre
- Le taux de passage par 0 (domaine temporel)
- Le flux entre « spectre d'amplitude »  $|\hat{S}|$  et « spectre d'amplitude reconstruit après liftrage (derniers coefficients mis à 0) »  $|\hat{S}'|$ , avec :

$$|\hat{S}'| = \left| \exp \left( \text{FFT} \left[ \text{Partie réelle} \left( \text{FFT}^{-1} [\log_e(|\hat{S}|)] \right) \underbrace{W}_{\text{liftre}} \right] \right) \right|$$

## TL – Partie 3b : Caractéristiques, exemple

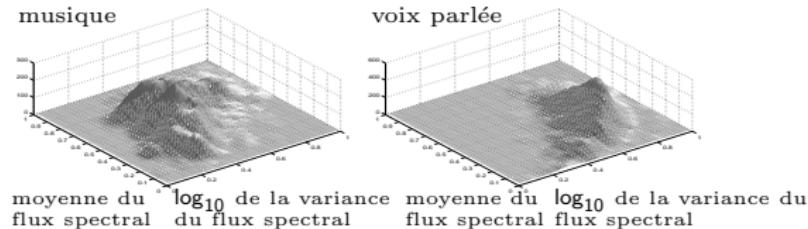
⇒ Densités de probabilité pour la caractéristique « flux spectral »  
à gauche : **musique** ; à droite : **voix parlée**



- **musique** : un seul mode
  - les caractéristiques de base restent relativement constantes sur un segment
- **voix parlée** : deux modes (le second plus rare que le premier)
  - les caractéristiques de base varient beaucoup sur un segment

## TL – Partie 3b : Fonctions d'observation

- **Moyenne** (espérance) par segment pour chaque Caractéristique
- $\log_{10}$  de la variance par segment pour chaque Caractéristique



- **Nombre de trames au-dessous de la moyenne** par segment pour chaque Caractéristique
- **Position x du max de l'histogramme** par segment pour chaque Caractéristique
- **Produit x par log<sub>10</sub> de la variance du mode correspondant** par segment pour chaque Caractéristique

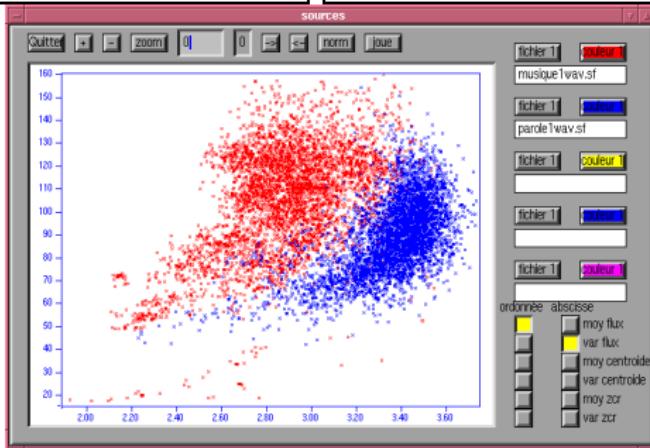
## TL – Partie 3b : Fonctions d'observation

### Autre représentation :

- Rouge : segments d'1 s de musique
- Bleu : segments d'1 s de voix parlée

### Fonctions d'observation :

- Abscisse :  $\log_{10}$  variance flux spectral
- Ordonnée : moyenne flux spectral



# Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ **Partie 5 : Décomposition d'un signal ("pursuit")**
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ Partie 7 : Conclusion

## L'idée

- ▶ On a un signal, venant d'un capteur
- ▶ On veut le décomposer en quelques éléments venant d'un dictionnaire  $D$  prédéfini (avec des poids)
- ▶ Par exemple : la TF fait ça : son dictionnaire, ce sont les exponentielles complexes (les poids sont les amplitudes, complexes possiblement : pour la phase)
- ▶ Autre exemple : les ondelettes font ça aussi ; le dictionnaire sont l'ondelette-mère et ses transformations (les poids sont les coefficients)
- ▶ On veut généraliser ça : on prend un dictionnaire redondant et vaste (comprenant les exponentielles complexes, des ondelettes, tout ce qu'on veut)

## Une solution : le matching pursuit (MP)

- ▶ Par contre, si  $D$  est vraiment grand, le problème n'est pas très abordable
  - ▶ en terme de temps de calcul, notamment : l'un des intérêts de la FFT, souvenez-vous, c'est d'être extrêmement rapide
- ▶ Le matching pursuit résoud ce problème (tout en restant glouton)

- ▶ Le signal s'écrit :  $f(t) \simeq f_N(t) = \sum_{n=1}^N a_n g_{\gamma_n}(t)$
- ▶ où chaque  $g_{\gamma_n}(t)$  est un élément (on dit un atome) du dictionnaire  $D$ , et  $a_n$  le poids associé

## Une solution : le matching pursuit

- ▶ L'idée est de récursivement trouver les  $N$  atomes de cette somme, de telle façon que l'atome choisi à une itération donnée est celui qui minimise l'erreur d'approximation (c'est-à-dire, qui maximise le produit scalaire avec le signal)
- ▶ Ainsi, à l'itération  $i + 1$ , le signal considéré est le signal à l'étape  $i$  auquel on a retiré l'atome trouvé à cette étape  $i$ , pondéré (par la valeur du produit scalaire)
- ▶ Note 1 : les atomes doivent être « normalisés »
- ▶ Note 2 : si on y regarde de près, on voit que si les atomes de  $D$  sont orthonormés, le MP (et ses variantes) sont une forme de PCA

## Une solution : le matching pursuit

- ▶ La solution itérativement trouvée est possiblement une solution sous-optimale :
  - ▶ il peut y avoir une solution  $f'_N(t)$  plus optimale, pour laquelle la différence entre  $f(t)$  et  $f'_N(t)$  soit plus petite que celle entre  $f(t)$  et  $f_N(t)$
  - ▶ ou, pour une différence entre  $f(t)$  et  $f_N(t)$  donnée, il pourrait exister une solution  $f_{N'}(t)$  pour laquelle on obtiendrait la même différence mais avec  $N' < N$

mais la solution optimale est introuvable dans un temps raisonnable, alors que la solution sous-optimale l'est

## Une solution : le matching pursuit

- ▶ Bien sûr, dans une décomposition de taille  $N$ , tous les atomes de  $D$  ne sont pas pris en compte : la taille de  $D$  est extrêmement plus grande que  $N$
- ▶ Il faut un critère d'arrêt : dès que pour un certain  $N$  la différence entre  $f(t)$  et  $f_N(t)$  devient inférieure à un seuil, l'algorithme itératif stoppe
- ▶ Donc : comme déjà dit,  $D$  peut être l'ensemble des exponentielles complexes utilisées par la TF
  - ▶ Mais, en ajoutant les ondelettes par exemple, on espère surpasser certaines des limitations de la TF (notamment en terme de résolution temporelle)
  - ▶ Et inversement, bien sûr : on espère que la TF surpassé certaines limitations des ondelettes

## Une solution : le matching pursuit ⇒ outils

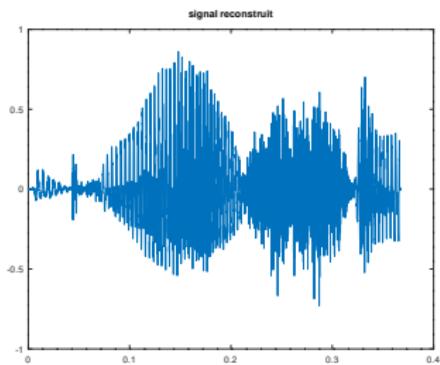
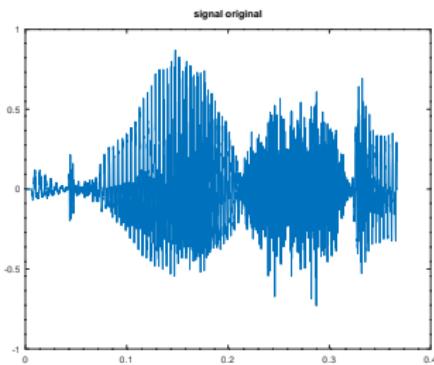
- ▶ Octave : “ltfat” ! Voir :  
<http://ltfat.github.io/doc/frames/franamp.html>
  - ▶ la fonction **franamp** n'est pas décrite dans la doc ltfat.pdf (!)
- ▶ Matlab :  
<https://fr.mathworks.com/help/wavelet/ref/wmpalg.html>
- ▶ Python : dans **scikit-learn**, il semble y avoir du Matching Pursuit

## franamp de “Itfat”

- ▶ On va tester la fonction de “Itfat” : voir **matching1.m**
- ▶ Paramètres :
  - ▶ ‘f’ : c'est un signal sonore (le mot 'greasy' dit par une femme)
  - ▶ ‘F’ (f majuscule) : c'est le dictionnaire ; ici, il contient ‘dgt’
    - ▶ ‘dgt’ : c'est la transformation de Gabor (soit, l'autre nom de la transformée de Fourier à court-terme, avec fenêtrage)
    - ▶ ‘hann’ : c'est la fenêtre utilisée (Hann=Hanning)
    - ▶ 256 : c'est le décalage entre 2 fenêtres
    - ▶ 1024 : c'est la taille de la FFT
    - ▶ note : 0-padding possible ???
  - ▶ ‘maxit’ : c'est le nombre max d'itérations
- ▶ Retours :
  - ▶ ‘c2’ : la décomposition obtenue (parcimonieuse)
  - ▶ ‘frec’ : le signal reconstruit
  - ▶ ‘info2’ : contient l'erreur de reconstruction, etc.

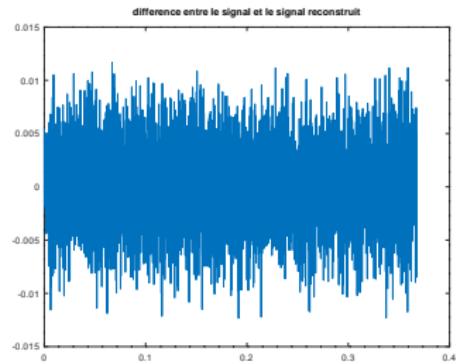
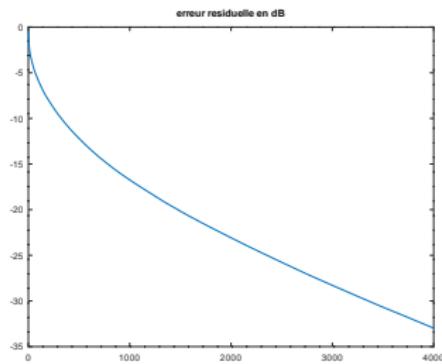
## franamp de “Itfat”

- ▶ Signal original et signal reconstruit (attention : ne garder que la partie réelle du signal reconstruit)



## franamp de “ltfat”

- ▶ Erreur en fonction de l'itération (en dB), et différence finale (après 4000 itérations) entre signal original et signal reconstruit



## franamp de “ltfat”

- ▶ “c2” : la décomposition obtenue (parcimonieuse)  $\Rightarrow$  il faut regarder ce qu'il y a dedans
  - ▶ taille du signal  $f$  : 5880 échantillons
  - ▶ taille de  $c2$  : 24576
  - ▶ **mais**, on s'est arrêté au bout de la 4000 itérations, ce qui veut dire qu'il n'y a pas plus de 4000 coefficients de  $c2$  (ce sont les poids des éléments du dictionnaire gardés) qui soient différents de 0
  - ▶ rappel : taille des trames : 1024
  - ▶ rappel : taille du décalage entre deux trames successives : 256

## franamp de “ltfat”

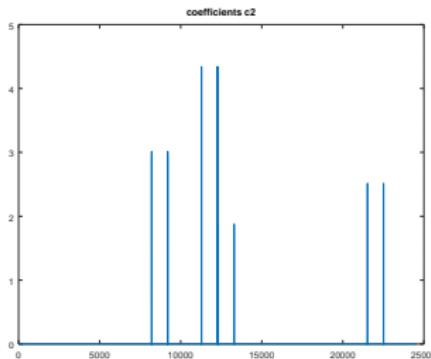
- ▶ Combien y-a-t'il de trames dans le signal ?
  - ▶  $24576/1024 = 24$  : il y a 24 trames  $\Rightarrow$  par contre,  $5880/256$  ne donne que 22.969 (**bizarre !**)
  - ▶ les 1024 premiers échantillons de “c2” concernent les 1024 premiers échantillons du signal (1 à 1024)  $\Rightarrow$  si on regarde on obtient bien un spectre du type FFT avec les fréquences réduites allant de 0 à  $1023/1024$
  - ▶ les 1024 échantillons de “c2” suivants concernent les échantillons du signal 257 à 1280
  - ▶ donc : la trame  $i$  concernent les échantillons de  $1 + 256 * (i - 1)$  à  $1 + 256 * (i - 1) + 1023$
  - ▶ et donc trame 23 : échantillons de 5633 à 6656 (6656 réduit à 5880, bien sûr)
  - ▶ et donc trame 24 : échantillons de 5889 (**pas possible !**) à ...
- ▶ **Bug probable** : pour la trame  $i$ , on prend les échantillons de  $1 + 255 * (i - 1)$  à  $1 + 255 * (i - 1) + 1023$

## franamp de “ltfat”

- ▶ Voir **matching2.m**, où je joue avec un signal enregistré par moi (je dis “parenthese”)
- ▶ Je fais varier “maxit” :
  - ▶ maxit = 10 ; ⇒ pas intelligible
  - ▶ maxit = 50 ; ⇒ presque intelligible
  - ▶ maxit = 100 ; ⇒ c'est déjà intelligible !!!
  - ▶ maxit = 500 ; ⇒ encore un peu “bizarre”
  - ▶ maxit = 2000 ; ⇒ quasi nickel
- ▶ “maxit” (c'est-à-dire  $N$ ) n'est pas plus grand que la taille du dictionnaire : pour “greasy”,  $D$  n'est pas de taille 1024, mais de taille  $24 \times 1024 = 24576$

## franamp de “ltfat”

- ▶ On peut voir ça en prenant “maxit = 10” et en plotant le module de “c2” :



⇒ Regardez comment sont disposés les coefficients gardés : par couples disposés symétriquement → typique de spectres de Fourier d'un signal numérique

## franamp de “ltfat”

- ▶ Mais, en fait, il est aussi possible que le même atome apparaisse plusieurs fois
- ▶ On va regarder le nombre de poids de “c2” supérieurs à 0, selon “maxit”

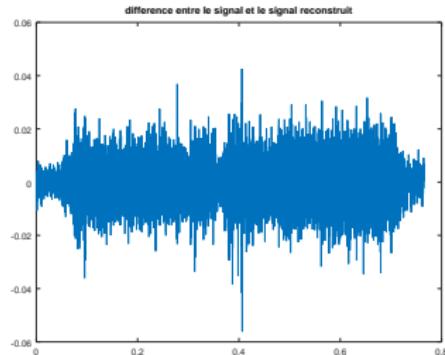
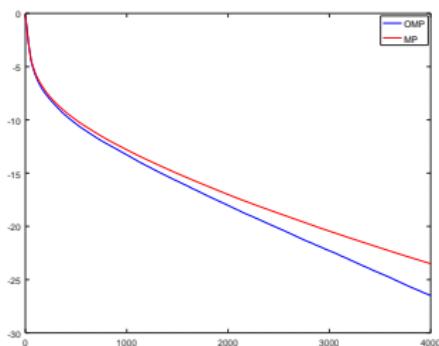
<i>maxit</i>	$c2 > 0$
10	10
50	50
100	100
500	498
2000	1985
4000	3937

## Une amélioration : le matching pursuit orthogonal (OMP)

- ▶ La différence par rapport au MP, c'est qu'à chaque itération, tous les poids  $a_n$  associés aux atomes sélectionnés jusqu'à présent (itération  $n$ ) sont réestimés, en projetant le signal sur l'ensemble de ces atomes (et pas seulement sur un seul, comme pour le MP)
- ▶ Normalement, on obtient des résultats meilleurs
  - ▶ c'est-à-dire, une erreur qui décroît plus vite
  - ▶ et donc, en fait, une parcimonie plus grande
- ▶ Du coup, par contre, ça demande plus de calculs, bien sûr

## franamp de “ltfat” – OMP versus MP

- ▶ Voir **matching3.m**
- ▶ Erreur en fonction de l'itération (en dB) pour l'OMP et le MP (bien sûr, ça se passe mieux pour l'OMP), et différence finale (après 4000 itérations) entre signal original et signal reconstruit pour l'OMP



## franamp de “ltfat” – OMP versus MP

- ▶ Est-il encore possible que le même atome apparaisse plusieurs fois ?
- ▶ Nombre de poids de “c1” plus grands que 0, selon “maxit”

<i>maxit</i>	$c_2 > 0$ (MP)	$c_1 > 0$ (OMP)	
10	10	10	
50	50	50	
100	100	100	
500	498	500	
2000	1985	2000	
4000	3937	4000	
6000	5843	6000	
7000	6769	7000	
7279	7023	7279	erreur min de l'OMP atteinte
7500	7227	7279	erreur min de l'OMP atteinte
8000	7685	7279	erreur min de l'OMP atteinte

## Une autre solution : la poursuite de base

- ▶ On change un peu de paradigme
- ▶ On retombe sur un problème d'optimisation, un peu du type de celui qu'on avait pour les SVM
- ▶ Il s'agit de minimiser  $\|x\|_1$ , tout en vérifiant  $Ax = b$
- ▶  $\|x\|_1$  est la norme  $l_1$  :  $\|x\|_1 = \sum_{i=1}^n |x_i|$
- ▶ Mais, en fait, si  $b$  est un signal à décomposer et  $A$  un dictionnaire, on retombe sur quelque chose qui ressemble au MP

## Une autre solution : la poursuite de base

- ▶ Tailles des différentes variables vues ci-dessus :
  - ▶  $b$  est un vecteur de taille  $m$
  - ▶  $A$  est de taille  $m \times n$  : la taille du dictionnaire est  $n$
  - ▶  $x$  est le vecteur des poids, de taille  $n$
- ▶ Bien sûr, tout n'est pas fini : idéalement, il faudrait qu'il y ait le moins possible de coefficients de  $x$  qui soient différents de 0, et la norme  $\ell_1$  ne permet pas d'assurer ça : il peut y avoir tout un tas de coefficients tous très petits, qui font que leur somme reste petite

## Une autre solution : la poursuite de base

- ▶ Pour la parcimonie, on veut annuler le plus de coefficients  $x_j$  possible et ça revient à résoudre le problème d'optimisation :
  - ▶ minimisation de  $\|x\|_0$ , tout en vérifiant  $Ax = b$
  - ▶  $\|x\|_0$  est le nombre d'éléments non nuls
- ▶ Par contre, ce problème là est NP-ardu  $\Rightarrow$  donc, on ne peut pas résoudre ça
- ▶ Le premier problème qu'on a vu (norme  $l_1$ ) est une **approximation raisonnable (et traitable)** du vrai problème (norme  $l_0$ ) qu'on a à résoudre

## Une autre solution : la poursuite de base

- ▶ Comment ça se fait ?
- ▶ Alors, pour les détails de la théorie, voir : S. S. Chen, D. L. Donoho et M. A. Saunders, "Atomic decomposition by basis pursuit", SIAM Journal on Optimization, vol. 20, 1998
  - ▶ disponible en ligne :

<http://web.stanford.edu/group/SOL/papers/BasisPursuit-SIGEST.pdf>

- ▶ **Raisonnable** : on peut prouver (mais c'est dur et long à faire) que la solution du premier problème a *tendance* à avoir beaucoup d'éléments nuls
- ▶ **Traitable** : ce premier problème est convexe, donc réécrivable comme un problème d'optimisation linéaire, donc soluble en temps polynomial

## Une autre solution : la poursuite de base

- ▶ La réécriture du problème initial, qui n'est pas linéaire à cause de la norme, sous forme d'un problème d'optimisation linéaire, est alors :
  - ▶ minimisation de  $e^T u + e^T v$
  - ▶ tout en vérifiant  $(A - A) \begin{pmatrix} u \\ v \end{pmatrix} = b$
  - ▶ et en vérifiant :  $u \geq 0, v \geq 0$
  - ▶ ce avec :  $x = u - v$
  - ▶ et avec :  $e$  un vecteur dont les composantes valent 1

## Une autre solution : la poursuite de base

- ▶ L'approche initiale utilise l'algorithme du simplexe ou de points intérieurs pour résoudre ce problème linéaire (voir votre cours d'optimisation)
- ▶ Quand les dimensions du problème sont trop grandes, il faut utiliser d'autres méthodes (algorithmes du premier ordre
  - “first-order methods” : FOMs)
    - ▶ il y a des paquets de FOMs !
    - ▶ voir :

[https://www2.isye.gatech.edu/~nemirovs/ActaFinal\\_2013.pdf](https://www2.isye.gatech.edu/~nemirovs/ActaFinal_2013.pdf)

- ▶ exemples (développés en détails dans l'article ci-dessus) :  
“composite minimization”, “saddle-point  $O(1/t)$ -converging first-order algorithms”

## La poursuite de base – Outils

- ▶ Octave/Matlab : “Itfat” contient des outils pour la poursuite de base
  - ▶ voir la fonction “franabp” (Frame Analysis Basis Pursuit)
- ▶ Python : il y a SPGL1 (Spectral Projected Gradient for L1 minimization), porté de Matlab
  - ▶ voir : <https://pypi.org/project/spgl1/>
- ▶ Note : pour le traitement du signal en général, dans tous les langages de programmation, ce site est peut-être utile/intéressant, pour poser des questions :  
<https://dsp.stackexchange.com/>

## La poursuite de base – Exemple

- ▶ Voir **basis2.m**, où je joue avec un signal enregistré par moi (je dis “parenthese”)
- ▶ Je fais varier “maxit” :
  - ▶ maxit = 10 ; ⇒ pas intelligible
  - ▶ maxit = 50 ; ⇒ pas très intelligible
  - ▶ maxit = 100 ; ⇒ c'est déjà intelligible !!!
  - ▶ maxit = 500 ; ⇒ encore un peu “bizarre”
  - ▶ maxit = 1000 ; ⇒ encore un peu “bizarre”
  - ▶ maxit = 2000 ; ⇒ quasi nickel
- ▶ ⇒ À comparer avec la matching pursuit.
- ▶ La taille du signal est de 12244 échantillons : on voit le taux de parcimonie qu'on peut atteindre

## Parcimonie – Résultats obtenus

- ▶ Voir un slide un peu plus haut :  $12244/500 = 24.488$
- ▶ Le critère que j'ai utilisé semble subjectif : je trouve à l'écoute que le son n'est plus qu'un peu "bizarre"
  - ▶ mais, en fait, dans la pratique (pour les sons) c'est comme ça qu'on fait : on fait écouter un grand nombre de sons plus ou moins compressés à beaucoup d'auditeurs, qui jugent la qualité de ces sons
- ▶ Un critère plus quantifiable (mais pas forcément plus objectif) : le rapport signal sur bruit (RSB), où le bruit est la différence entre le signal original et le signal compressé
  - ▶ utilisable pour les signaux autres qu'audio et/ou vidéo

## Parcimonie – Temps de calcul : attention

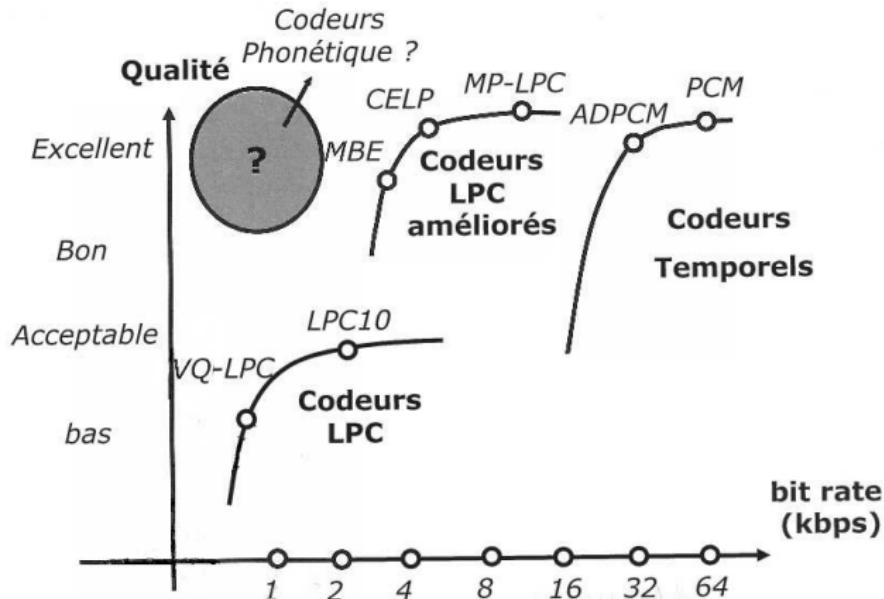
Sur un son de 2,6 secondes, échantillonné à 16 kHz, j'obtiens ça :

maxit	MP	OMP	BP
10	0.214038 s	0.631746 s	2.609758 s
50	0.947096 s	3.594427 s	2.651748 s
100	1.829337 s	7.187554 s	2.688686 s
200	3.555159 s	15.237985 s	2.666716 s
400	7.795347 s	31.523382 s	2.649987 s
800	14.885222 s	63.588161 s	2.638233 s
2000	37.903205 s	154.488862 s	2.658571 s
	$\simeq \text{maxit}^{0.97}$	$\simeq \text{maxit}^{1.05}$	cste

## Parcimonie – Exemple : la parole (téléphonie)

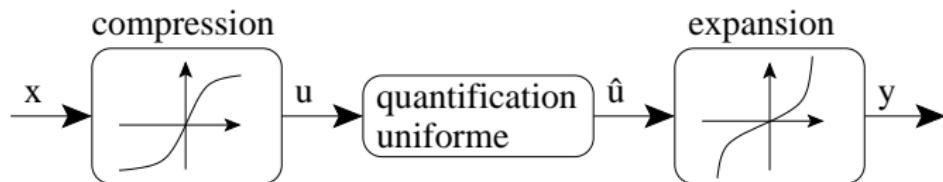
- ▶ Il s'agit de coder la parole de telle manière que le débit soit le plus bas possible
- ▶ Hypothèse : la parole est une séquence de phonèmes (sons)
  - ▶ 8-10 phonèmes/seconde
  - ▶ 30-50 phonèmes pour une langue :  $64 = 2^6 \Rightarrow 6$  bits
  - ▶ donc le débit minimal devrait se situer autour de  $6 \times 10 = 60$  bps
  - ▶  $\Rightarrow$  il faut comparer ça aux 128 kbps pour un .wav de parole ( $16000 \times 8$ ), aux 64 kbps sur une ligne numérique (PCM), (aux 33.6 kbps pour une ligne téléphonique analogique)
- ▶ Note : la parole, c'est le contenu ci-dessus, **mais** ça contient aussi de l'info. concernant le locuteur, de l'info. paralinguistique...  $\Rightarrow$  100 bps semble être la limite
- ▶ **parcimonie** : diviser par  $\simeq 1000$  semble être la limite extrême

## Parcimonie – Exemple : la parole (téléphonie)



## Exemple : la parole (téléphonie) : quantification

- ▶ Quantification non-uniforme ( $\simeq$  passage .wav à PCM)



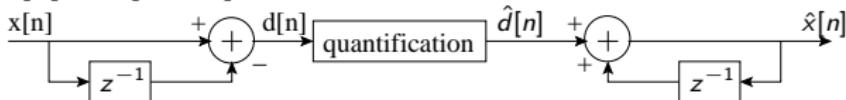
- ▶ Europe : loi A

$$\begin{aligned} &\blacktriangleright \frac{A|x|}{1 + \ln(A)} \operatorname{sgn}(x) && \text{pour } 0 \leq \frac{|x|}{x_{\max}} \leq \frac{1}{A} \\ &\blacktriangleright x_{\max} \frac{1 + \ln\left(\frac{A|x|}{x_{\max}}\right)}{1 + \ln(A)} \operatorname{sgn}(x) && \text{pour } \frac{1}{A} \leq \frac{|x|}{x_{\max}} \leq 1 \end{aligned}$$

- ▶ exemple : avec 8 bits en loi A on obtient le même RSB (rapport signal sur bruit) qu'avec 12 bits en uniforme

## Exemple : la parole (téléphonie) : DPCM – Differential Pulse Modulation Coding

- ▶ Le signal de parole est redondant : chaque échantillon peut être ± prédit à partir des échantillons précédents ; notamment, si on *spécule* à l'ordre 1, on peut quantifier la dérivée numérique :  $d[n] = x[n] - x[n - 1]$ , plutôt que directement le signal



- ▶ Et, si l'autocorrélation de  $x$  à l'ordre 1 est supérieure à  $0.5\sigma_x^2$  (on *spécule*), alors la variance de  $d$  est inférieure à celle de  $x$  ; en effet :  $\sigma_d^2 = 2\sigma_x^2 - 2R_{xx}[1]$

- ▶ Et alors le RSB augmente de :

$$\Delta \text{RSB} = 20 \log_{10} \left( \frac{\sigma_x}{\sigma_d} \right) = 10 \log_{10} \left( \frac{1}{2(1 - R_{xx}[1]/\sigma_x^2)} \right)$$

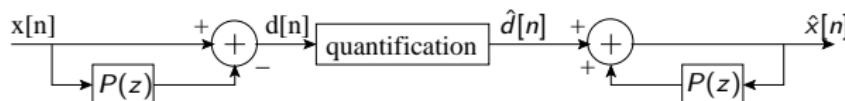
- ▶ À performance égale en RSB on peut réduire  $b$  le nombre de bits

## Exemple : la parole (téléphonie) : DPCM – Plus généralement

- Maintenant, on tente de prédire le signal avec plusieurs échantillons passés :

$$d[n] = x[n] - \sum_{i=1}^p a_i x[n-i]$$

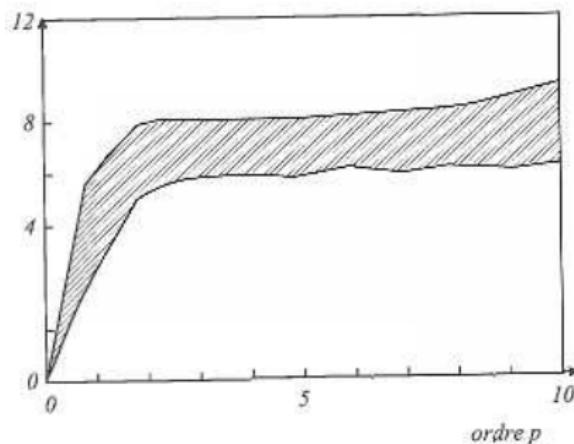
- À la place de  $z^{-1}$ , on a :  $P(z) = \sum_{i=1}^p a_i z^{-i}$



- Note : on se retrouve donc dans un cas ressemblant à de la prédiction linéaire (LPC/modélisation AutoRégressive (AR))

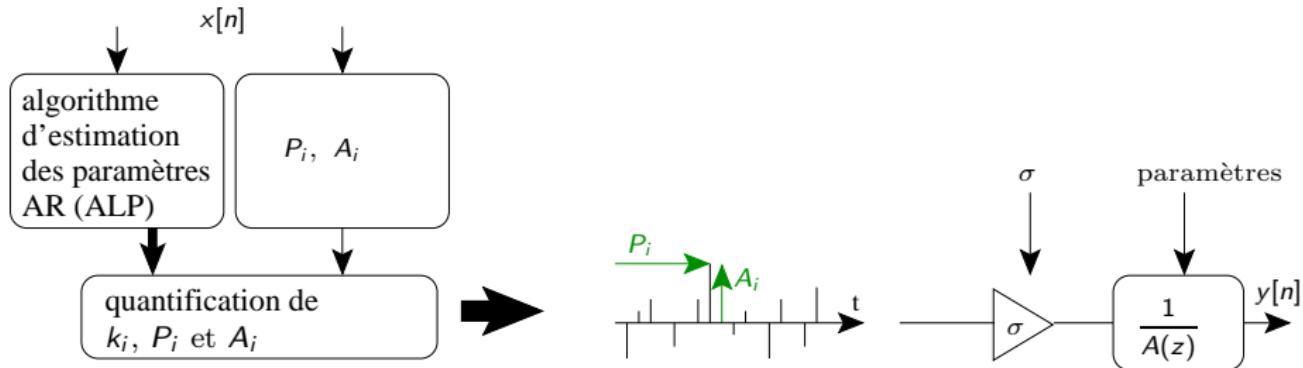
## Exemple : téléphonie : DPCM – Gain de prédiction

- ▶ Le rapport  $\Delta\text{RSB} = 20 \log_{10} \left( \frac{\sigma_x}{\sigma_d} \right)$  est appelé le gain de prédiction
- ▶ On cherche l'ordre  $p$  optimum :  $\Delta\text{RSB}$  doit être grand et  $p$  petit
- ▶ Sur une grande base de données, on voit que  $p = 4$  suffit :



- ▶ ADPCM : Adaptive DPCM

## Exemple : la parole (téléphonie) : Codeurs MP-LPC

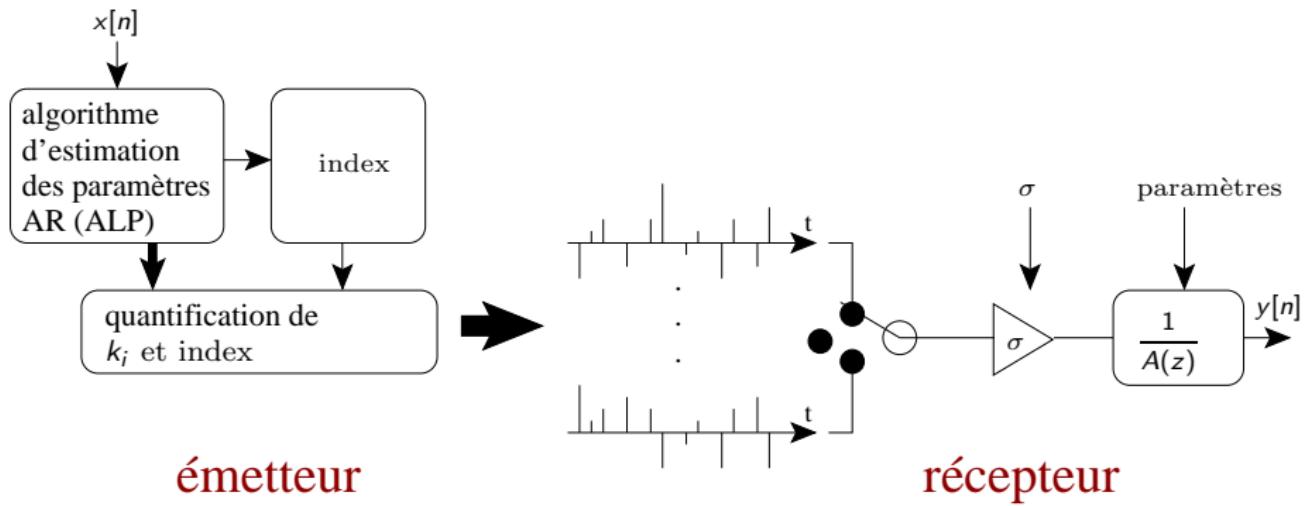


émetteur

récepteur

- ▶ GSM (RPE) : 13000 bps

Exemple : la parole (téléphonie) : Codeurs MP-LPC (CELP)



- GSM (2ème génération) : 4800 bps

## Parcimonie – Exemple : signaux “fixes” (enregistrés)

- ▶ Codage/Stockage
  - ▶ représenter toute l'information avec un minimum de ressources (codage sans perte ; Huffman...)
  - ▶ théorie de l'information pour aller plus loin (TdS, ML, ...)
- ▶ Compression/Stockage
  - ▶ il faut comprendre ce qui est pertinent et ce qui ne l'est pas, notamment perceptuellement (codage avec perte)
  - ▶ images : jpg versus bmp/gif (taux de compression  $\simeq 20/25$ )
  - ▶ vidéos : mpg versus avi (taux de compression  $\simeq 30$ )
  - ▶ sons (parole, et **musique** : plus compliquée que la parole !) : mp3 versus wav (taux de compression  $\simeq 10$ )

## TL – Partie 4 : compression audio

- ▶ Comparer MP, OMP et BP sur un son enregistré par vous
  - ▶ une petite phrase (de quelques mots)
  - ▶ pas trop longue (2 ou 3 secondes), sinon le temps de calcul devient trop long
- ▶ Attention : utiliser les mêmes “frames” dans les trois cas
  - ▶ essayez d’autres “frames” que Gabor et Hanning (comme j’ai fait)
- ▶ Attention à la fréquence d’échantillonnage (44100 n’est pas nécessaire)
- ▶ Comparer les erreurs finales pour les trois “pursuit”

# Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ **Partie 6 : Analyse en composantes indépendantes**
- ▶ Partie 7 : Conclusion

## Notions d'entropie

- ▶ L'entropie de Shannon est une mesure de la quantité d'information contenue ou délivrée par une source d'information : texte écrit, signal électrique, fichier informatique...
- ▶ D'abord, le calcul de l'entropie d'une source d'infos donne une mesure de l'info minimale qu'on doit coder pour représenter ces infos sans perte
- ▶ Et, ensuite, dans le cas particulier de la compression de fichiers, l'entropie indique le nombre minimal de bits que peut atteindre un fichier compressé
- ▶ En pratique, l'entropie d'une image ou d'un son peut être abaissée en retirant des détails imperceptibles pour les humains, comme lors de la compression des sons par le format MP3, des images par JPEG ou des vidéos par MPEG

## Notions d'entropie – Formellement

- ▶ Pour une source qui est une variable aléatoire discrète  $X$  comportant  $n$  symboles, chaque symbole  $x_i$  ayant une probabilité  $P_i$  d'apparaître, l'entropie  $H$  de la source  $X$  est définie comme

$$H_b(X) = -\mathbb{E} [\log_b P(X)] = \sum_{i=1}^n P_i \log_b \left( \frac{1}{P_i} \right) = - \sum_{i=1}^n P_i \log_b(P_i)$$

- ▶ où  $\mathbb{E}$  est l'espérance mathématique et  $\log_b$  le logarithme de base  $b$
- ▶ On utilise en général un logarithme à base 2 (l'entropie possède alors les unités de bit/symbole) ; et finalement :

$$H(X) = H_2(X) = - \sum_{i=1}^n P_i \log_2(P_i)$$

## Notions d'entropie – Exemple

- ▶ On considère une langue naturelle qui comprend ces 8 sons :  
[a], [b], [c], [d], [e], [f], [g], [h]
- ▶ Pour coder ces 8 sons, il faut 3 bits, car  $2^3 = 8$
- ▶ Mais, là, on fait l'hypothèse que tous les sons sont équiprobables :  $P_i = \frac{1}{8} = 0.125 \quad \forall i$
- ▶ Ce qui donne une entropie de :  $H(X) = -8 \times \frac{1}{8} \log_2 \left( \frac{1}{8} \right) = 3$   
(bien sûr : ce sont nos 3 bits, comme prévu)
- ▶ Mais, dans une langue naturelle, tous les sons ne sont pas équiprobables

## Notions d'entropie – Exemple

- ▶ Si on prend cette configuration, pour les probabilités :  
 $P = [0.50 \ 0.29 \ 0.06 \ 0.05 \ 0.04 \ 0.03 \ 0.02 \ 0.01]$
- ▶ On obtient comme entropie :  $H(X) = 1.994371$
- ▶ **C'est-à-dire qu'on peut espérer coder les sons de notre langue naturelle sur seulement 2 bits au lieu de 3**
- ▶ En fait, on peut montrer que la longueur moyenne en bits  $L$  est comprise dans cet intervalle :  $H(X) \leq L < H(X) + 1$

## Notions d'entropie – Exemple

- ▶ On regarde ce qui se passe dans notre cas. Codons chaque son ainsi :

son	code binaire	longueur $l_i$	$P_i$
[a]	0	1	0.50
[b]	10	2	0.29
[c]	110	3	0.06
[d]	1110	4	0.05
[e]	11110	5	0.04
[f]	111110	6	0.03
[g]	1111110	7	0.02
[h]	1111111	7	0.01

- ▶ note : quand on rencontre un 0, on sait qu'on a la fin d'un son ; sauf pour le dernier : sept 1 d'affilée indique qu'on est en présence d'un [h] ⇒ il est facile de se convaincre que cette manière de faire ne prête jamais à confusion (il n'y a jamais deux manières possibles d'interpréter une suite de 0 et de 1)

## Notions d'entropie – Exemple

- ▶ La longueur moyenne est alors de :  $L = \sum_{i=1}^n l_i P_i$ , soit ici : 2.05
- ▶ Exemple : [c][a][g][e] se code ainsi : 1100111111011110
- ▶ On reçoit cette suite de 0 et 1 :

1000101001111101111101001111110100010011111110101000  
01111110111011111000111110001100010000111101001111010  
11101010001100100010101000100110100101001010110011010  
00011001110101001011110010101011011010000010000100

- ▶ Ce qui donne :  
baabbagfbagbaabahabbaaagdfaafaacaabaaaebaebdbbaaca-  
baabbbaabacbababbacacbaaacadbabeabbbccbaaaabaaaba
- ▶ Voir “entropie.m” : on trouve des  $L$  proches de 2.05

## Débit d'entropie d'un signal aléatoire bruité

- ▶ Si le signal est bruité, il faut plus de bits pour assurer que le message soit correctement reçu
  - ▶ notamment, il faut que le message soit redondant (la même info doit être envoyée plusieurs fois, pour être sûr de ne pas faire d'erreur)
- ▶ C'est-à-dire que l'entropie du signal, à cause du bruit, a augmenté
  - ▶ si  $p$  est la probabilité qu'un 0 se transforme en 1 (et inversement), on peut montrer que l'entropie du signal augmente d'un facteur :

$$\frac{1}{1 + p \log_2(p) + (1 - p) \log_2(1 - p)}$$

- ▶ pour  $p = 0.01$  (1%), le facteur vaut 1.0879 ;  $p = 0.05 \Rightarrow 1.4013$  ;  $p = 0.2 \Rightarrow 3.5962$

## Information mutuelle

- ▶ L'information mutuelle de deux variables aléatoires est une quantité mesurant la dépendance statistique de ces variables ; elle se mesure en bit
- ▶ Deux variables sont indépendantes si la réalisation de l'une n'apporte aucune information sur la réalisation de l'autre
- ▶ Le coefficient de corrélation est une mesure du cas particulier de dépendance dans lequel la relation entre les deux variables est strictement linéaire
- ▶ L'information mutuelle est nulle si et seulement si les variables sont indépendantes, et croît lorsque la dépendance augmente

## Information mutuelle – Formellement

- ▶ Soit  $(X_1, X_2)$  un couple de variables aléatoires de densité de probabilité jointe égale à  $P(x_1, x_2)$
- ▶ Et soient les densité de probabilité marginales  $P(x_1)$  et  $P(x_2)$
- ▶ Alors l'information mutuelle est dans le cas discret :

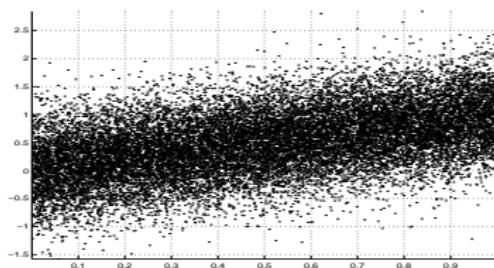
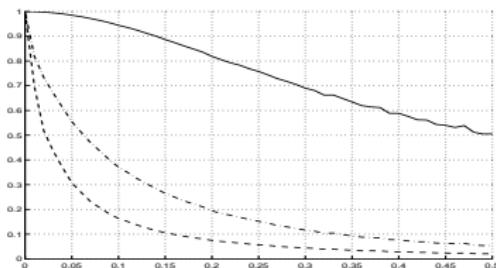
$$I(X_1; X_2) = \sum_{x_1} \sum_{x_2} P(x_1, x_2) \log \frac{P(x_1, x_2)}{P(x_1) P(x_2)}$$

et, dans le cas continu :

$$I(X_1; X_2) = \int_{\mathbb{R}} \int_{\mathbb{R}} P(x_1, x_2) \log \frac{P(x_1, x_2)}{P(x_1) P(x_2)} dx_1 dx_2$$

## IM – Exemple 1 : dépendance linéaire : $x_2 = x_1 + y$

- $x_1$  est uniformément répartie entre 0 et 1 et  $y$  est normale, de variance  $\sigma^2$  et de moyenne nulle

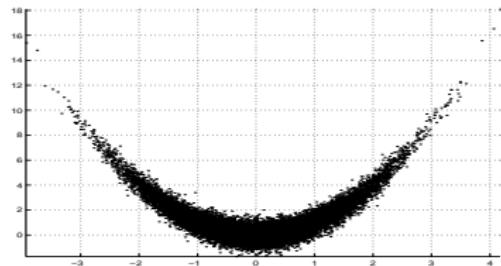
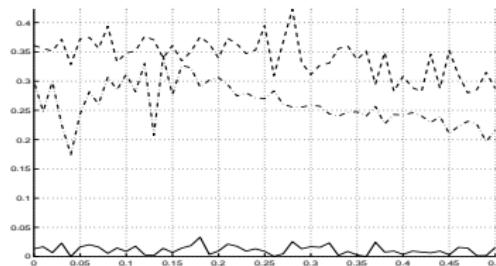


Abscisse :  $\sigma$  y ; ordonnée : (— coef. Observation avec  $\sigma = 0,5$ ; abscisse : corrélation), (--- IM), (- -  $\chi^2$ )       $x_1$ , ordonnée :  $x_2$

- ▶ Le coeff. de corrélation, l'IM normalisée  $im(x_1, x_2)$  et le  $\chi^2$  sont égaux à 1 quand  $x_2 = x_1$
  - ▶  $im(x_1, x_2)$  diminue plus vite que le coeff. de corrélation, et le test du  $\chi^2$  plus vite encore, au fur et à mesure que  $x_1$  et  $x_2$  se décorrèlent

IM – Exemple 2 : dépendance parabolique :  $x_2 = x_1^2 + y$

- ▶  $x_1$  et  $y$  sont normales, de variances 1 et  $\sigma^2$  et de moyennes nulles

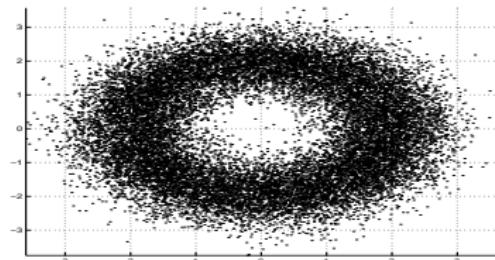
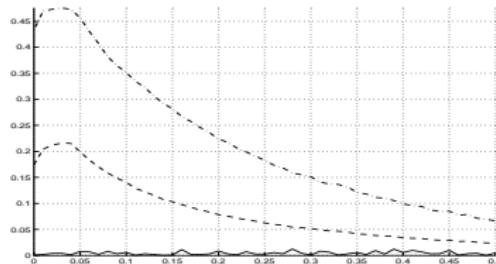


Abscisse :  $\sigma$  y ; ordonnée : (— coef. Observation avec  $\sigma = 0,5$ ; abscisse : corrélation), (--- IM), (- -  $\chi^2$ )       $x_1$ , ordonnée :  $x_2$

- ▶ Nous voyons que le coefficient de corrélation n'est pas du tout efficace dans ce cas, contrairement à l'information mutuelle et au test du  $\chi^2$

## IM – Exemple 3 : dépendance circulaire

- ▶  $y$  est uniformément répartie entre 0 et  $2\pi$ ;  $x_1$  est égale à  $2\cos(y) + z_1$  et  $x_2$  à  $2\sin(y) + z_2$ ;  $z_1$  et  $z_2$  sont gaussiennes de variances  $\sigma^2$  et de moyennes nulles

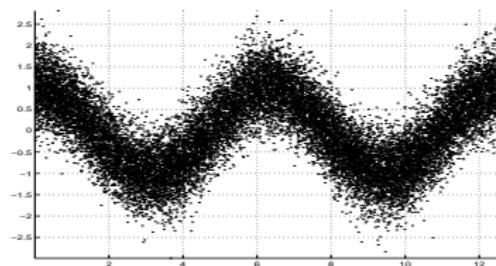
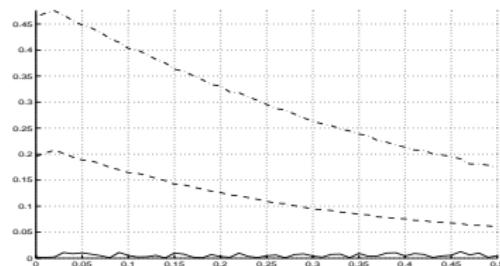


Abscisse :  $\sigma z_1/z_2$ ; ordonnée : (— coeff. Observation, avec  $\sigma = 0,5$ ; abscisse : corrélation), (-.- IM), (- -  $\chi^2$ )  
ordonnée :  $x_1$ , ordonnée :  $x_2$

- ▶ On voit que le coeff. de corrélation n'est pas du tout efficace dans ce cas, contrairement au test du  $\chi^2$  mais surtout à l'information mutuelle

## IM – Exemple 4 : dépendance sinusoïdale

- ▶  $x_1$  est uniformément répartie entre 0 et  $4\pi$ ;  $x_2$  est égale à  $\cos(x_1) + y$ ;  $y$  est gaussienne de variance  $\sigma^2$  et de moyenne nulle

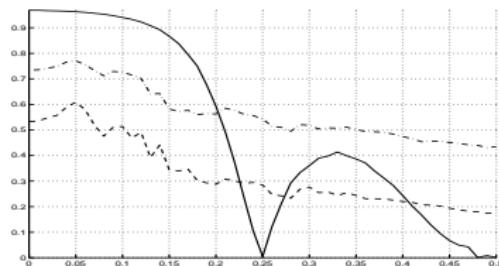


Abscisse :  $\sigma_y$ ; ordonnée : (— coef. Observation, avec  $\sigma = 0,5$ ; abscisse : corrélation), (--- IM), (- -  $\chi^2$ )                     $x_1$ , ordonnée :  $x_2$

- ▶ On voit que le coefficient de corrélation n'est pas du tout efficace dans ce cas, contrairement au test du  $\chi^2$  mais surtout à l'information mutuelle

## IM – Exemple 5 : dépendance en arc de cercle

- ▶  $y$  est uniformément répartie entre 0 et  $2\alpha\pi$ ,  $\alpha$  variant entre 0.01 et 0.5 ;  $x_1$  est égale à  $2 \cos(y)$  et  $x_2$  à  $2 \sin(y)$



Abscisse :  $\alpha$ ; ordonnée : (— coef. Observation avec  $\alpha = 0.375$  ; abscisse : corrélation), (-.- IM), (- -  $\chi^2$ )

- ▶ On voit que les performances du coefficient de corrélation se détériorent (on passe par un 0 de corrélation pour  $\alpha=0,25$ ), contrairement à celles de l'information mutuelle et du test du  $\chi^2$ , qui restent à peu près stables

## Information mutuelle – Lien avec l'entropie

- ▶ On a :

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X, Y) \end{aligned}$$

- ▶  $H(X)$  et  $H(Y)$  sont les entropies de  $X$  et  $Y$ ,  $H(X|Y)$  et  $H(Y|X)$  les entropies conditionnelles, et  $H(X, Y)$  l'entropie conjointe entre  $X$  et  $Y$
- ▶  $H(X, Y)$  : si chaque paire d'états possibles  $(x, y)$  des V.A.  $(X, Y)$  a une probabilité  $p_{x,y}$  alors l'entropie conjointe est :

$$H(X, Y) = - \sum_x \sum_y p_{x,y} \log_2(p_{x,y})$$

- ▶ Et :  $H(Y|X) \equiv H(X, Y) - H(X)$

## Information mutuelle – Lien avec Kullback-Leibler

- ▶ On cherche à déterminer une distance entre deux densités de probabilité  $P$  et  $Q$
- ▶ La définition de la distance/divergence de Kullback-Leibler, dans le cas général, est :

$$D_{\text{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

- ▶ On a, dès lors :

$$I(X_1; X_2) = KL(P(x_1, x_2), P(x_1)P(x_2))$$

$$= \sum_{x_1} \sum_{x_2} P(x_1, x_2) \log \frac{P(x_1, x_2)}{P(x_1)P(x_2)}$$

- ▶ Si le log qu'on prend est à base 2, on obtient des bits

## Analyse en composantes indépendantes

- ▶ Au départ, l'ACI a été développée pour la séparation aveugle de sources, et plus particulièrement pour résoudre le problème de la soirée cocktail (cocktail party problem)
  - ▶ on dispose de  $P$  microphones dans une salle dense, où  $N$  personnes discutent par groupes de tailles diverses
  - ▶ et on est dans le cas où  $P > N$  (il y a plus de microphones que de personnes)
  - ▶ note : c'est une partie du sujet du projet de ST7 qu'une partie d'entre vous environ a à traiter (sauf que vous, la localisation vous aide : pas ici)
- ▶ Puis, l'ACI a été utilisée pour d'autres problèmes :  
[http://www2.iap.fr/users/cardoso/compsep\\_planck.html](http://www2.iap.fr/users/cardoso/compsep_planck.html)

## Analyse en composantes indépendantes – Modèles

### 1. Modèle linéaire instantané non bruité : $x = As$

- ▶  $s$  sont les sources ; il y en a  $n$
- ▶  $x$  sont les signaux qu'on observe ; il y en a  $p$
- ▶  $A$  est la matrice de mélange ; elle est de taille  $p \times n$
- ▶ ⇒ et bien sûr il y a une dimension supplémentaire : le temps

$$\begin{pmatrix} x_1(t) \\ \vdots \\ x_p(t) \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{p1} & \dots & a_{pn} \end{pmatrix} \begin{pmatrix} s_1(t) \\ \vdots \\ s_n(t) \end{pmatrix}$$

- ▶ conditions pour pouvoir résoudre ce problème :
  - ▶ une seule source suit une distribution gaussienne (bruit)
  - ▶ le rang de  $A$  doit être égal au nombre de sources
- ▶ note : modèle pas très réaliste pour de vrais problèmes (par exemple : mélange de sources sonores)

## Analyse en composantes indépendantes – Modèles

2. Modèle linéaire instantané bruité :  $x = As + \sigma$
3. Modèle linéaire non instantané :
  - ▶ le mélange peut être convolutif  $\Rightarrow$  c'est le cas pour le mélange de sources sonores, où l'effet de la réverbération de salle est vu comme un filtrage dans le cadre des SLI classiques (la sortie est égale à l'entrée convoluée par la réponse impulsionnelle du SLI)
  - ▶ la plupart du temps, on peut s'en sortir en passant dans le domaine de Fourier (des fréquences, en général)  $\Rightarrow$  alors, le produit de convolution étant transformé en produit simple, on peut se ramener à un problème linéaire (bruité bien sûr)
4. Mélanges non linéaires :  $x = F(s)$ , où  $F(\cdot)$  est une fonction non linéaire quelconque ; pas de méthode générale dans ce cas  $\Rightarrow$  des méthodes existent pour des cas particuliers (mais dans le cas le plus général, le problème est mal posé, et la solution n'est pas unique (domaine de recherche actif))

## Analyse en composantes indépendantes – Algorithmes

Les algorithmes les plus connus/les plus utilisés sont (mais il y en une énorme quantité; un survey très général peut être trouvé à cet endroit : <http://users.ics.aalto.fi/aapo/papers/NCS99web/>) :

- ▶ Algorithme HJ (pour Hérault-Jutten). Je ne vais donner des détails que sur lui ci-dessous; je vais le développer un peu dans le cas où  $n = p = 2$ .
- ▶ Maximisation de Contraste (CoM)
- ▶ Joint Approximate Diagonalization of Eigenmatrices (JADE)
  - ▶ à ce sujet, voir l'excellent site  
[http://www2.iap.fr/users/cardoso/compsep\\_classic.html](http://www2.iap.fr/users/cardoso/compsep_classic.html)  
où aussi tout un tas de code est donné (fait par des spécialistes/inventeurs de l'ICA : Cardoso, Souloumiac, Jutten)
- ▶ Fast-ICA
- ▶ Infomax

## Analyse en composantes indépendantes – HJ

- ▶ c'est un algorithme *neuro-mimétique*, et itératif
- ▶ le problème se réécrit ainsi :  $y = Wx = WAs$  où  $W$  est une matrice de séparation, dont tous les termes diagonaux sont nuls
- ▶ l'estimation des sources se fait alors ainsi :  $y = (I + W)^{-1}x$  avec comme règle d'adaptation (d'une itération à la suivante)  $\Delta w_{ij} = f(y_i)g(y_j)$
- ▶ où  $f(\cdot)$  et  $g(\cdot)$  sont des fonctions, différentes, et dont au moins une est non linéaire impaire  $\Rightarrow$  **le cœur de la méthode se trouve dans ces 2 fonctions !**
  - ▶ la justification théorique pour l'utilisation de ces fonctions non linéaires est intervenue plus tard

## Analyse en composantes indépendantes – HJ

- ▶ On va regarder un peu plus en détails l'algorithme HJ et on va le tester, dans le cas  $n = p = 2$  linéaire non bruité
- ▶ On s'intéresse à la séparation d'un mélange linéaire instantané, où on a deux sources et deux capteurs (problème tel que formulé par Jutten)
- ▶ On utilise un critère de déconvolution non linéaire pour exploiter l'indépendance des entrées
- ▶ On considère le problème de Hérault-Jutten présenté ci-dessous, où les observations  $y_1$  et  $y_2$  sont des combinaisons linéaires des sources  $u_1$  et  $u_2$  (inconnues) ; soit :

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

## Analyse en composantes indépendantes – HJ

- ▶ On suppose que la matrice de mélange  $A = (a_{ij})_{i,j}$  (inconnue) est régulière et que les sources  $u_1$  et  $u_2$  sont stationnaires, indépendantes, de moyennes nulles (soit  $E(u_1) = E(u_2) = 0$ , hypothèse fréquente en séparation de sources)
- ▶ Les sorties  $\hat{u}_1$  et  $\hat{u}_2$  du séparateur, estimations respectives des sources  $u_1$  et  $u_2$ , vérifient :

$$\begin{cases} \hat{u}_1 &= y_1 - w_{12}\hat{u}_2 \\ \hat{u}_2 &= y_2 - w_{21}\hat{u}_1 \end{cases}$$

où  $w_{12}$  et  $w_{21}$  sont les poids adaptatifs du séparateur, qui doivent être déterminés de sorte que les 2 conditions suivantes soient vérifiées :

- ▶ chaque sortie  $\hat{u}_i$  est proportionnelle à une seule source  $u_i$ ;
- ▶ les sorties  $\hat{u}_1$  et  $\hat{u}_2$  sont indépendantes

## Analyse en composantes indépendantes – HJ

- ▶ En remplaçant  $y_1$  et  $y_2$  par leurs expressions en fonction de  $u_1$  et  $u_2$ , on obtient ceci :

$$\begin{cases} \hat{u}_1 = a_{11}u_1 + a_{12}u_2 - w_{12}\hat{u}_2 \\ \hat{u}_2 = a_{21}u_1 + a_{22}u_2 - w_{21}\hat{u}_1 \end{cases}$$

- ▶ On peut exprimer  $\hat{u}_1$  et  $\hat{u}_2$  en fonction des sources  $u_1$  et  $u_2$  :

$$\begin{cases} \hat{u}_1 = \frac{1}{1-w_{12}w_{21}}((a_{11}-w_{12}a_{21})u_1 + (a_{12}-w_{12}a_{22})u_2) \\ \hat{u}_2 = \frac{1}{1-w_{12}w_{21}}((a_{21}-w_{21}a_{11})u_1 + (a_{22}-w_{21}a_{12})u_2) \end{cases}$$

- ▶ L'indépendance des signaux  $u_1$  et  $u_2$  implique qu'ils sont décorrélés, soit  $E(u_1u_2) = 0$ , et donc  $E(\hat{u}_1\hat{u}_2) = 0$
- ▶ Cependant, la décorrélation n'est pas suffisante pour trouver les  $w_{12}$  et  $w_{21}$
- ▶ De plus, l'indépendance des sorties  $\hat{u}_1$  et  $\hat{u}_2$  implique que pour toutes fonctions  $f$  et  $g$  arbitraires, on a  $E(f(\hat{u}_1)g(\hat{u}_2)) = 0$

## Analyse en composantes indépendantes – HJ

- ▶ On peut prendre, par exemple, les fonctions suivantes :  
 $f : u \rightarrow u^3$  et  $g : u \rightarrow u$
- ▶ À partir de ces résultats, Hérault et Jutten ont proposé de déterminer les coefficients  $w_{12}$  et  $w_{21}$  de sorte que :

$$\begin{cases} E(\hat{u}_1^3 \hat{u}_2) = 0 \\ E(\hat{u}_2^3 \hat{u}_1) = 0 \end{cases}$$

- ▶ Ils proposent donc pour  $w_{12}$  et  $w_{21}$  la loi d'adaptation :

$$\frac{\delta w_{ij}}{\delta t} = \mu \hat{u}_i^3 \hat{u}_j$$

où  $\mu$  est une constante positive

- ▶ Et on obtient un algorithme itératif estimant  $w_{12}$  et  $w_{21}$
- ▶ Et, plus important, on obtient finalement un algorithme itératif estimant  $\hat{u}_1$  et  $\hat{u}_2$

## ACI – Test de Hérault-Jutten

- ▶ Voir **jutten1.m**
- ▶ Les deux signaux qu'on mélange et qu'on essaie de démêler sont deux signaux harmoniques
- ▶ La fréquence fondamentale de chacun d'eux est tirée aléatoirement entre 50 et 1000 Hz
- ▶ Le nombre de partiels harmoniques est tiré aléatoirement
- ▶ Les amplitudes des partiels sont tirées aléatoirement
- ▶ Et les deux paramètres anti-diagonaux de la matrice de mélange sont eux aussi tirés aléatoirement
- ▶ Puis on vérifie si l'algorithme arrive à démêler les deux signaux

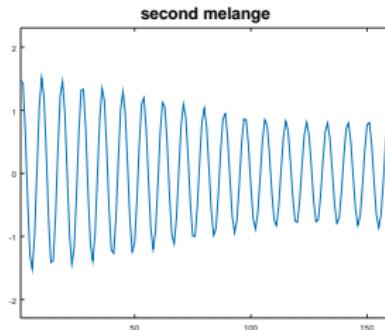
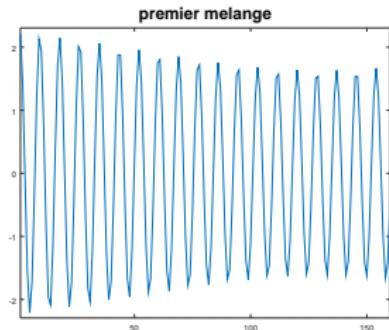
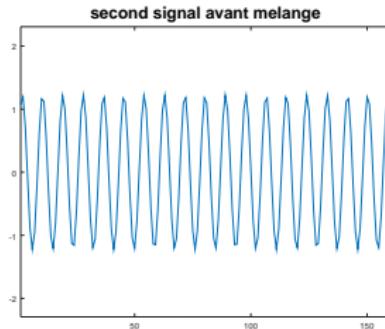
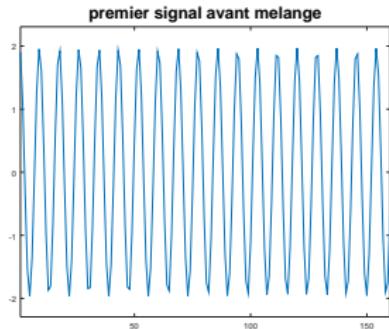
## ACI – Test de Hérault-Jutten : voir **jutten1.m**

- ▶ Conditions d'arrêt de l'algorithme :
  - ▶ on abandonne après  $n$  itérations maximum ( $n$  grand)
  - ▶ ou alors on arrête dès que le  $\frac{\delta w_{ij}}{\delta t}$  entre deux itérations successives devient inférieur à  $1e^{-7}$
- ▶ Note : il faut aussi déterminer si  $\hat{u}_1$  correspond à  $u_1$  et  $\hat{u}_2$  correspond à  $u_2$ , ou l'inverse
- ▶ On obtient ces performances (sur beaucoup d'essais) :

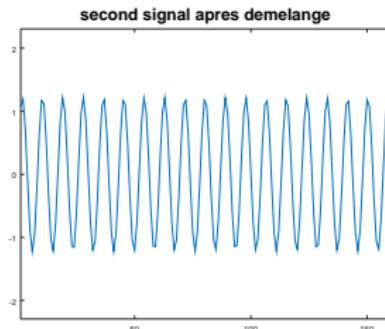
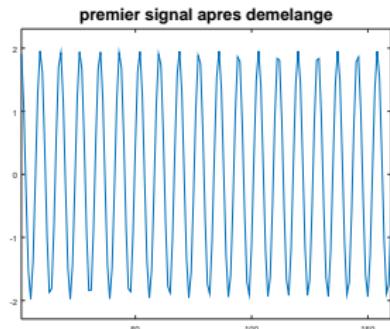
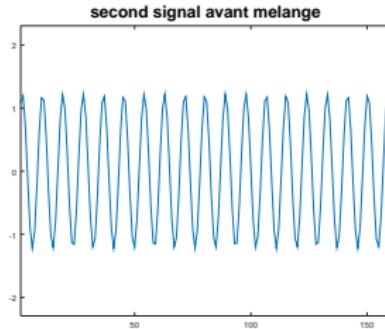
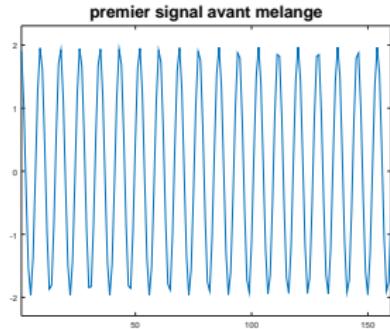
$n$	10000	20000	30000	40000	50000	60000	70000	80000	200000
succès en %	495/1000 49,5	348/633 55	291/483 60,2	224/359 62,4	295/492 60	213/353 60,3	169/271 62,4	177/269 65,8	189/255 74,1

- ▶ Attention : le temps de calcul peut vite devenir prohibitif
- ▶ Attention : réglage de certains paramètres par fait

$$\text{HJ-}a_{12} = 0.27586, a_{21} = 0.21411, f_1 = 938.44, f_2 = 915.04$$



$$\text{HJ-}a_{12} = 0.27586, a_{21} = 0.21411, f_1 = 938.44, f_2 = 915.04$$



## Déconvolution aveugle

- ▶ Le problème est de déterminer la solution  $e$  d'une équation de la forme :  $s = e * h$ 
  - ▶ où  $s$  est la mesure
  - ▶ où  $e$  est le signal qu'on veut restaurer (inconnu)
  - ▶ et  $h$  est la réponse impulsionnelle perturbatrice (connue ; ou pas)
- ▶ Note : en traitement des images,  $h$  est souvent nommée fonction d'étalement du point (point spread function ou PSF en anglais), plutôt que réponse impulsionnelle
- ▶ Dans le cas de vrais processus physiques, il y a bien sûr du bruit (inconnu) en plus :  $s = e * h + \sigma \Rightarrow$  la déconvolution directe, si  $h$  est connue, n'est pas possible, du coup

## Déconvolution aveugle – $h$ connue

Il y a deux familles de méthodes pour faire la déconvolution aveugle dans ce cas :

- ▶ Le filtrage de Wiener
- ▶ La méthode de Richardson-Lucy

On va un peu regarder la première pour  $h$  connue

Note : on fait aussi l'hypothèse qu'on a une idée de la DSP du bruit (souvent, il est blanc)

## Déconvolution aveugle – $h$ connue – Filtrage de Wiener

- ▶ L'idée est de trouver un  $g(t)$  raisonnable tel que  $\hat{e}(t) = g(t) * s(t)$  soit une estimation de  $e$  qui minimise l'erreur quadratique moyenne
- ▶ Quand on écrit tout dans le domaine fréquentiel, on obtient :

$$G(f) = \frac{1}{H(f)} \left[ \frac{|H(f)|^2}{|H(f)|^2 + \frac{1}{\text{SNR}(f)}} \right] = \frac{1}{H(f)} \left[ \frac{|H(f)|^2}{|H(f)|^2 + \frac{\sigma(f)}{S(f)}} \right]$$

- ▶ et ça s'interprète facilement : pour les fréquences où le bruit est relativement petit le terme entre crochets est proche de 1 et le filtre de Wiener est l'inverse du système ; pour les fréquences où le bruit augmente le RSB diminue et le terme entre crochets diminue (le filtre de Wiener atténue les composantes fréquentiels trop bruitées)

## Déconvolution aveugle – Outils

- ▶ Dans matlab, il y a : “deconvwnr”

<https://fr.mathworks.com/help/images/ref/deconvwnr.html>

qui fait la déconvolution aveugle d'image, pour le défloutage, basé sur le filtre de Wiener

- ▶ Dans matlab, il y a aussi : “deconvlucy”

<https://fr.mathworks.com/help/images/ref/deconvlucy.html>

qui fait la même chose avec l'algorithme de Lucy-Richardson

- ▶ Pour la déconvolution aveugle proprement dite on a : “deconvblind”

<https://fr.mathworks.com/help/images/deblurring-images-using-the-blind-deconvolution-algorithm.html>

- ▶ Pour octave on trouve les mêmes fonctions, téléchargeables ici :

<http://savannah.gnu.org/patch/?8571>

(téléchargez “octave-contrib.tgz”, qu'on trouve en bas de page dans la rubrique “Documents joints”)

## Déconvolution aveugle – $h$ inconnue

C'est la déconvolution aveugle proprement dite. Algorithmes :

- ▶ Basés sur les moments d'ordre supérieurs du signal
- ▶ Ou basés sur des versions modifiées de l'algorithme de Lucy-Richardson original pour lequel  $h$  est connu
- ▶ Alors, Lucy-Richardson, pour  $h$  connu est une méthode itérative ; et à l'itération  $i + 1$  on a :

$$\hat{e}_{i+1}(t) = \left( \frac{s(t)}{\hat{e}_i(t) * h(t)} * h(-t) \right) \hat{e}_i(t)$$

## Déconvolution aveugle – $h$ inconnue

- ▶ Et Lucy-Richardson, pour  $h$  inconnue, est une méthode doublement itérative :

$$\hat{h}_{i+1}(t) = \left( \frac{s(t)}{\hat{h}_i(t) * \hat{e}_i(t)} * \hat{e}_i(-t) \right) \hat{h}_i(t)$$
$$\hat{e}_{i+1} = \left( \frac{s(t)}{\hat{e}_i(t) * \hat{h}_i(t)} * \hat{h}_i(-t) \right) \hat{e}_i(t)$$

- ▶ Note : c'est la méthode utilisée par matlab

# Plan

- ▶ Partie 1 : Introduction (qu'est-ce que la parcimonie ?)
- ▶ Partie 2 : Analyse harmonique (plus de choses sur la TF, etc.)
- ▶ Partie 3 : Analyse multirésolution (ondelettes, etc.)
- ▶ Partie 4 : Notions d'apprentissage supervisé
- ▶ Partie 5 : Décomposition d'un signal ("pursuit")
- ▶ Partie 6 : Analyse en composantes indépendantes
- ▶ **Partie 7 : Conclusion**

# Conclusion

- ▶ On a vu deux familles de transformations : la TFCT et les ondelettes ; elles permettent d'obtenir des représentations parcimonieuses des signaux
- ▶ On a vu comment les mélanger avec la “pursuit” pour améliorer encore la parcimonie
- ▶ Il y a d'autres familles de transformations :
  - ▶ adaptées aux signaux à analyser
  - ▶ ou valables a priori pour tous les signaux

# Conclusion

- ▶ Il y a d'autres familles de transformations :
  - ▶ adaptées aux signaux à analyser : par exemple, pour la parole, on extrait ce qu'on appelle les formants, notamment grâce aux méthodes d'analyse spectrale haute-résolution du type "modèles AR(MA)"
  - ▶ ou valables a priori pour tous les signaux :
    - ▶ voir par exemple les méthodes qui appartiennent à ce qu'on appelle la "classe de Cohen" (par exemple Wigner-Ville)  $\Rightarrow$  on trouve certaines de ces méthodes dans les outils "Itfat" et "tftb" (regardez les docs)
    - ▶ et il y en a d'autres encore : MUSIC/ESPRIT, Pisarenko, Prony, etc.

# Conclusion

- ▶ Ainsi, pour la parole par exemple (quelque chose de similaire est valable pour tous les signaux), on arrive à diviser la taille des signaux par 30 environ, alors qu'en théorie, au vu de l'information contenue, on devrait pouvoir la diviser par 1000 environ
  - ▶ on a fait la moitié du chemin
  - ▶ par contre, on ne sait pas vraiment comment faire l'autre moitié, là